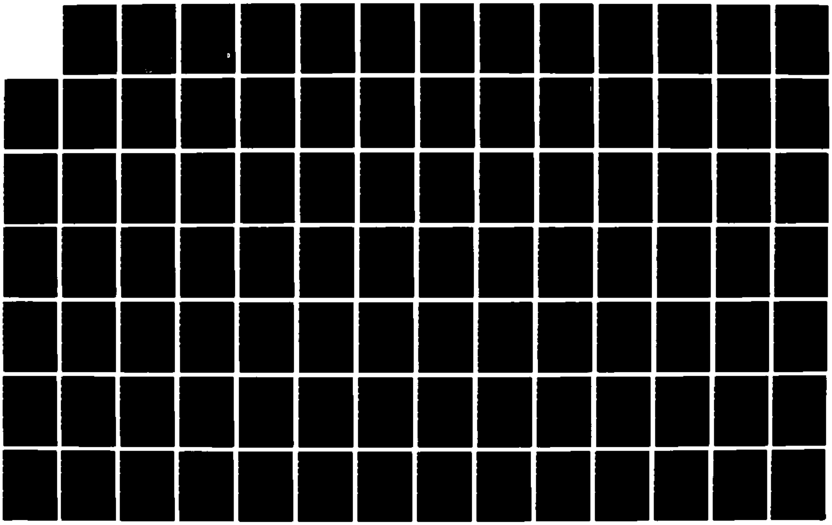
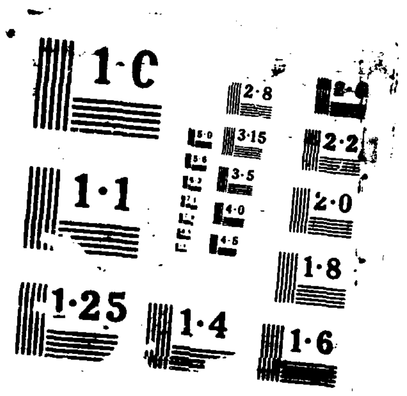
AD-A193 119 DEVELOPMENT OF A HYBRID SIMULATOR FOR ROBOTIC MANIPULATORS(U) AIR FORCE INST OF TECH WRIGHT-PATTERSON AFB OH SCHOOL OF ENGINEERING P M VAN WIRT DEC 87 1/2

UNCLASSIFIED AFIT/GE/ENG/87D-68 F/G 12/5 NL




1·0
2·8
2·5
5·0
5·6
3·15
3·5
2·2
4·0
2·0
4·5
1·1
1·8
1·25
1·4
1·6

DEVELOPMENT OF A HYBRID SIMULATOR FOR ROBOTIC MANIPULATORS

THESIS

Peter M. Van Wirt
Captain, USAF

AFIT/GE/ENG/87D-68

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY
AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

1

AFIT/GE/ENG/87D-68

DEVELOPMENT OF A HYBRID SIMULATOR FOR ROBOTIC MANIPULATORS

THESIS

Peter M. Van Wirt
Captain, USAF

AFIT/GE/ENG/87D-68

AFIT/GE/ENG/87D-68

DEVELOPMENT OF A HYBRID SIMULATOR FOR ROBOTIC MANIPULATORS

THESIS

Presented to the Faculty of the School of Engineering

of the Air Force Institute of Technology

Air University

In Partial Fulfillment of the

Requirements for the Degree of

Master of Science in Electrical Engineering

Peter Madison Van Wirt, B.S.

Captain, USAF

December 1987

Approved for public release; distribution unlimited

Preface

The purpose of this thesis is to develop and validate the capability to conduct real time research in hybrid control on a robot manipulator simulation. A side benefit is the ability to conduct man-in-the-loop simulations in real time.

The simulator model parameters are identified through use of previous research in robot control, and by experimental determination of nonconservative forces. Once available, the model is programmed into the analog portion of a SIMSTAR hybrid computer. The simulation is validated by comparison of error profiles with actual errors produced on the PUMA 560. There are many future research topics that can be developed from this base, mostly in the area of controls research. To this researchers knowledge, it is,to date, the only real time robot simulation capable of testing hybrid control schemes.

I would like to express my appreciation to several individuals, without whom I could not have completed my research. My thesis advisor, Captain Michael Leahy, provided his considerable expertise in the area of robot simulation/control research. His help is instrumental to developing the proper model to be implemented. I would like to also acknowledge the help I received from Captain Mikel Miller in collecting data, running the PUMA, and editing my thesis. Finally, I would like to thank my wife, Debbie, daughter Stephanie, and son Michael for supporting my effort and for being understanding when the time crunches came.

Peter M. Van Wirt

Table of Contents

Page

List of Figures

List of Tables

AFIT/GE/ENG/87D-68

Abstract

A real-time robot manipulator simulation capability has been developed. By programming the robot dynamics in the analog section of a SIMSTAR Hybrid Computer, the computational burden of digital integration techniques is avoided, and due to the analog nature of the model, the simulation can be run in real time without sacrificing accuracy. The ability to test analog and hybrid control schemes is also achieved through the development of an analog manipulator model on the SIMSTAR and because the SIMSTAR is both a digital and an analog computer. A hybrid controller contains an analog feedback portion to provide needed loop stiffness, and a digital feedforward portion to compensate for the changing dynamics of a robotic manipulator. The model is developed through a combination of previous research and experimental evaluation. Once programmed, the SIMSTAR model is validated using known trajectory/error data obtained from the AFIT PUMA 560. (Theses).

Chapter 1

INTRODUCTION

Motivation

In the future, Air Force planners envision robots capable of meeting many of the Air Force's needs for ground maintenance in an increasingly hostile environment. The flight line of the future will be populated by teleoperated and autonomous robots. It will be necessary for these robots to duplicate the speed, range of motion, and payload of the human arm. To meet these needs, robots will have to improve their performance in many areas [1,7,8].

One of the most basic improvements will be in the area of manipulator control. Current industrial control schemes are inadequate for most flight line tasks [7]. Industrial robots rely on a high gain feedback loop to reject disturbances, and make no adaptations for changes in manipulator dynamics. They lack the speed and accuracy that will be necessary to quickly handle munitions and perform complex maintenance tasks.

Future robot controllers will be based on non-linear techniques capable of calculating the dominant dynamics of the manipulator, and then compensating the control inputs for the dynamics of the robot. Because these new controllers compensate for robot dynamics, they are known as dynamics based controllers. Due to the inherent complexity, their design and evaluation is a complicated task.

Both feedforward compensation and the computed-torque algorithm provide the adaptation necessary for future controllers

[7]. These controllers are being compared to standard industrial controllers in terms of mid-course and end point errors [1]. The end point errors of the standard industrial controllers are smaller, due largely to the fact that the poles chosen for the standard industrial controller are much stiffer than those chosen for the computed-torque and feedforward controller. The industrial controller is stiffer because industrial robots are not easily reconfigurable, and high sampling rates are required for the non-linear controllers to attain stiff feedback loops [5:172].

Dynamics based control may be implemented digitally or in a hybrid analog/digital combination. Hybrid controllers combine the stiffness of analog Proportional/Derivative (PD) loops with the dynamic compensation of the dynamics based controllers. Current robot simulations do not allow real-time testing of controllers. Digital implementations of these simulators do not allow evaluation of hybrid controllers. Both of these restrictions not only limit control simulation studies, but also preclude simulation of man-in-the-loop control of vertically articulated robots, which will play an important role in any flight-line application. The motivation for this thesis is the development of a hybrid control simulation system capable of testing digital controllers and digital/analog controllers.

Objective

Objective comparison of different controllers in a uniform environment is difficult because current robots are ill suited for experimental evaluation of modern control techniques. Simulation of a manipulator can provide this capability by allowing the controller to be implemented without the restrictions imposed by existing hardware. Simulation studies can only provide reliable comparison if the model used is validated. The objective of this thesis is to develop and validate an analog model of a PUMA 560 and an environment that allows both digital, analog, and hybrid control of that model.

Problem Statement

Existing industrial robot controllers lack the ability to strongly reject mid course errors. End point errors are minimized with the controllers, but the Proportional/Derivative (PD) loop, which treats all changes in dynamics as disturbances, guarantees relatively high mid course errors.

One limitation to research in dynamics based control of robots is the inability to separate errors introduced by mismodeling the dynamics of a manipulator and errors introduced through an inability to stiffen the controller due to sampling rate inadequacies. Because existing commercial systems are difficult to modify for dynamics based control, the inability to sample at high rates introduces errors in trajectory tracking. Previous research has identified most of the significant terms in the dynamics of the Puma manipulators [10]. The true indication of controller accuracy has not been obtained because the controller testing environments have not provided adequate computational capacity to allow the sampling rates necessary to aggressively suppress errors in the feedback loop.

The classic approach to that problem is to conduct simulation studies. Simulation environments for geared manipulators have been developed by several individuals. Leahy developed a complete simulation/testing environment for the PUMA 600 [6]. This required a redesign of the existing PUMA control scheme. For the first time the control engineer could simulate a controller on a digital computer, and then test the same controller on an industrial manipulator. Problems encountered included a considerable restriction on the pole placement for any PD controller that required dynamic calculations, and inadequate modelling of motor dynamics.

To overcome the problems associated with comparing controllers at sampling rates that are not adequate for the particular controller, the feedback portion of the controller could be implemented in analog. Simulation of hybrid controllers requires an analog simulator. In this thesis the analog model of a PUMA 560 is coded in the analog section of a SIMSTAR Hybrid Computer.

An analog simulator avoids many of the problems present in digital simulations. Digital simulation must withstand many approximations forced on it by the digital computer. The approximations made in mathematical functions such as integration creates more computational burden than the simulation would otherwise require. The analog macros(components) compute in real time, thus avoiding errors created by having only discrete values, although some macros (such as trigonometric functions) are approximations.

In a digital computer, the computational requirements of both the controller and the simulation play an important role in developing a simulation/testing environment. Limitations on a computer's ability to provide this computational capacity can restrict the success of the simulation/testing environment.

Current robot simulations can not support real-time testing of controllers. Real time digital simulation requires more computer speed than just a digital robot controller because, in addition to the controller, the computer must generate the robot model. In order to produce accurate results, the sampling rate of the digital model of the manipulator is expected to exceed the sampling rate of the controller by a factor of 10. In the case of manipulator controllers, a sampling rate of 2 ms is considered adequate to hold positive control over the arm movements [5]. This would suggest the model sampling rate required would be 200 microseconds. Adding to the problem is that accurate digital simulation of a robot requires a fourth order Runga-Kutta integration which calculates arm dynamics four times during each sample interval. This would then require dynamics calculations every 50 microseconds. The complexity of the manipulator dynamics makes this sampling rate difficult, if not impossible, to achieve in real-time. Even if the simulation is performed off-line, the computational load is still great.

A high speed computer could reduce the computational problems, but digital implementations of robot simulators do not allow testing hybrid controllers. A hybrid controller has a digital feedforward portion and an analog feedback portion. Inability to simulate in analog, and in real-time, precludes simulation of man-in-the-loop control of vertically articulated robots, which will play an important role in any flight-line application.

These problems cloud the issues surrounding dynamics based control.- The issue of which terms in the manipulator dynamics are significant from a control engineer's point of view can be clarified when differing complexity controllers are compared in terms of trajectory tracking capability. This trajectory tracking capability becomes unclear when each controller requires different PD pole placement or each controller is operated under the constraints imposed on the most complex controller due to computational constraints.

Method of Approach

The problem encountered in the digital implementation of a simulation environment can be overcome by approaching the modeling problem in the context of analog computing. Unfortunately, most analog simulations would not be easily suited to testing digital control algorithms. To help overcome the problems with digital simulations, this thesis develops, from previous sources and experimental data, an accurate model of a industrial manipulator. This model will be programmed into the analog portion of a SIMSTAR hybrid computer in a manner that supports testing of both digital, analog, and digital/analog controllers.

The first step in producing an accurate simulation is to ensure that the mathematical model that is used accurately reflects the manipulator dynamics. Due to the complex nature of robot dynamics, it is important to make this model as compact and efficient as possible, without unduly affecting the model's accuracy. The PUMA 560 is used as a case study because it is a a six degree of freedom vertically articulated industrial manipulator, representative of a type required for many Air Force applications and experimentally generated error datais available. This thesis will use Tarn's simplified equations for PUMA 560 dynamics as a starting point for the model [11].

All vertically articulated robot structures require a drive system with high torque capability. The PUMA derives this high torque through a high gear ratio gear train. One advantage of the high gear ratio of the PUMA is that the magnitude of many link dynamic forces become insignificant. These terms, made up of coriolis and centrifugal coupling terms, can be eliminated from the model without adverse affect.

Another effect of the PUMA's high gear ratio is to make the motor dynamics a critical component of overall system dynamics. The two components of motor dynamics necessary for an accurate model are actuator inertia and viscous friction. Actuator inertia magnitudes for PUMA have been identified in previous research [10]. To completely model the actuator, step input data will be collected from the PUMA and a model for the actuator viscous friction will be fit to this data. Motor dynamics will be added to Tarn's link dynamics model to provide an accurate model for the simulation.

The dynamic model of the PUMA 560 will then be programmed into the analog portion of the SIMSTAR hybrid computer. The AFIT SIMSTAR consists of two different computers. The Parallel Simulation Portion(PSP) contains analog macros that can simulate dynamics in real-time. The Digital Analog Processor(DAP) is the digital computer that runs in parallel with the PSP for a particular simulation. This processor will contain the digital portion of any controller being tested.

Once the simulation is in place in the SIMSTAR, it must be validated. Validation is performed by determining the simulation's ability to accurately predict the error trend information for a particular controller. This is accomplished by using a trajectory that has been previously tested on a robot with a particular controller. The joint position error data generated by experimental evaluation is then compared to the simulated error data generated, with the same controller, on the SIMSTAR. The simulation will be considered validated when the error trends generated by the simulation reflect the actual errors generated on the manipulator. Trend information is defined as error direction and approximate error magnitude.

Limitations

The most severe limitation placed on this research are due to the limited number of analog macros currently available on the SIMSTAR hybrid computer. The number of analog macros limits the number of multipliers and summers available in the Parallel Simulation Portion(PSP). The number of analog macros directly effects the number of joints that can be effectively modelled, as well as the particular form of equations that can be implemented.

The decision is made to strive for simulation accuracy over simulating the full six degrees of freedom. To allow a more accurate model, only the first three joints of the PUMA are modelled. These joints are the positioning joints of the PUMA, and the last three joints control the orientation of the end effector. The position joints are more susceptible to dynamic changes than the orientation joints, so are more appropriate to dynamics based controls research. The orientation joints are modelled as a static load at the end of link three. It is important to note that the coupling between the positioning and orientation joints is quite weak, increasing our ability to describe the first three joints effectively.

Even with only three links modelled, the number of analog macros available is still insufficient to model every dynamic component. The SIMSTAR simulation is currently able to model only the significant terms of the dynamics equations. This is the main reason for finding the significant terms in the PUMA 560 dynamics equations, for both digital and analog modelling. In a digital model, the problem is linked to sampling rates whereas, in an analog model the problem is again the number of analog components available to model the joints.

Another limitation, imposed by the SIMSTAR, is that the Digital to Analog(D/A) and the Analog to Digital(A/D) conversion rates are tied to the loop rate in the digital portion of the SIMSTAR. Also, the device used for D/A and A/D conversion is limited because the interrupt routines overhead time restricts how fast the digital portion can execute. Combined, these restrictions bound the highest sampling rate that can be implemented at 7 ms. This would be a strict limitation, but for the fact that this thesis intends to implement the PD loop in analog. The feedforward portion of the controller is implemented digitally, and its output is sent to the analog portion every 7 ms., along with the desired velocity and position.

One basic limitation of any simulation is the inexact knowledge of what is being modelled. Unmodelled effects in the SIMSTAR create a limitation on the ability of the simulation to truly reflect the reaction of the PUMA 560 to a particular control scheme. The PUMA 560 exhibits vibration in some cases, depending on the sampling rate and complexity of the controller. This effect is noted, for a particular controller, when exercised from some initial conditions, but not from others. This effect is caused by sampling rate problems but the exact nature of the effect is not known. The inability to model this effect limits the ability to identify a minimum sampling rate needed for a particular controller.

Contribution

The implementation of an analog model of the PUMA 560's positioning joints on the SIMSTAR hybrid computer permits further identification of important controller components. The analog model allows real-time simulation for both controller research and man-in-the-loop investigations. Also, a stepping stone has been provided to allow integration and experimental evaluation of a digital/analog controller, on the SIMSTAR, with the PUMA 560 for actual testing.

Organization

The thesis is organized in the following manner. Chapter 2 reviews previous research done in the manipulator control, and manipulator simulation. Chapter 3 contains the model development and implementation. Chapter 4 contains simulation verification results. Chapter 5 presents conclusions and recommendations.

Chapter Two

Background

This chapter reviews prior research in the area of dynamics based robotic control, with particular attention to work done using geared manipulators similar to the PUMA 560 Robot Arm used in this research.

Robot control has received considerable attention from the robotic community in recent years, due to the constraints that present controllers place on industrial robots.

> "Similarly, the PUMA is controlled to the same peak speeds and accelerations independent of load. These peaks, then, bound the maximum load that can be carried while maintaining acceptable tracking characteristics. Again, this constitutes a performance limitation due to the control law"[2:1].

A great deal of research has been conducted on the application of modern control techniques on both geared manipulators, and direct-drive manipulators [2,5,7,8]. Robot control is a complicated issue, due to the nonlinear, coupled dynamics of the manipulator. Each manipulator type requires separate consideration due to the impact the high gear ratios have on the significance of terms in the dynamics equations. "When gearing is eliminated, however, the full nonlinear dynamic interactions between moving links are manifested"[1:165]. Geared manipulator dynamics require more attention to motor dynamics because the gearing does reduce the impact of link dynamics. The research to date has explored robot dynamics, limitations of present controllers, actuator dynamics, and modern control techniques [4,11].

DYNAMICS

Robot dynamics can be described in terms of x non-linear, coupled, differential equations.

$$nT_m = n^2 J_m \ddot{\theta} + n^2 B_m \dot{\theta} + T_f + T_l \qquad (2.1)$$

where:

x	- number of joints
n	- x by x diagonal matrix of gear ratios
T_m	- x by 1 vector of motor torque
J_m	- x by x diagonal matrix of actuator inertia
B_m	- x by x diagonal matrix of actuator viscous friction
T_f	- x by 1 vector of static friction torque
T_l	- x by 1 vector of load torque, made up of inertial terms, coriolis and centrifugal terms, and gravity

Load torques are commonly represented by Lagrange-Euler dynamics equations. The Lagrange-Euler equation of motion for the robot can be described as:

$$T_l = D(\ddot{\theta})\theta + h(\theta,\dot{\theta}) + g(\theta) \qquad (2.2)$$

where

T_l	- x by 1 vector of joint torques
$\theta,\dot{\theta},\ddot{\theta}$	- x by 1 vectors of joint position, velocity, and acceleration
$D(\theta)$	- x by x inertia matrix
$h(\theta,\dot{\theta})$	- x by 1 vector of coriolis and centrifugal terms
$g(\theta)$	- x by 1 vector of gravity terms

This equation is computationally intense, especially when x=6 (the full six degrees of freedom of the PUMA arm). The complete Lagrange-Euler formulation for a six D.O.F. manipulator involves 66,271 multiplications and 51,548 additions[4:732]. "A major stumbling block in the drive for real-time implementation has been the computational complexity of these formulations" [7:23]. For this reason much time and effort has been spent identifying the significant terms in the equation. Insignificant terms are then removed leaving a simplified set of equations without adversely impacting the model accuracy. Then, symbolic algorithms are used to simplify the remaining equations [7:51].

The dynamics can also be described in terms of Neuton-Euler (N-E) dynamics equations. The N-E algorithm significantly reduces the computations required. The Neuton-Euler formulation for 6 DOF requires 852 multiplications and 738 additions[4:732]. However, the N-E formulation is not easily interpreted to determine the significance nature of joint coupling [4].

Contemporary industrial controllers assume linear decoupled dynamics, and rely on the inherent disturbance rejection capabilities of their controllers to cope with disturbances produced by unmodelled dynamics. Industrial controllers use a very basic proportional, derivative(PD) feedback law to control errors from a desired trajectory. The poles of this closed loop system are chosen based on worst case dynamics configuration to ensure stability throughout the operating envelope. The pole placement is chosen to guarantee stability, but the actual poles are only critically damped at maximum load [6:66]. Unfortunately, in order to maintain adequate performance and stability, the operating velocities are restricted by the controller, not the hardware. This class of controllers assumes that the arm dynamics can be linearized and decoupled, leaving only a simple second order dynamic model for each link. That model is further simplified by the assumption that the self inertia is a constant. Goor states that robot performance is constrained by the standard industrial controller because the controller causes speed and load dependant errors [3:387]. The mathematical formulation for the controller is:

$$T(t) = K_v(\dot{\theta}_d - \dot{\theta}) + K_p(\theta_d - \theta) \qquad (2.3)$$

where:

$T(t)$	- drive torque
K_v, K_p	- controller gain coefficients
$\dot{\theta}_d, \theta_d$	- desired velocity and position
$\dot{\theta}, \theta$	- actual joint velocity and position

The equation used to identify K_v and K_p is:

$$A(S^2 + 2yw_nS + w_n^2)e = 0 \qquad (2.4)$$

where:

A	- constant
y	- damping ratio
w_n	- undamped natural frequency
e	- error

In this case, A is the self inertia term for the joint to be controlled. The damping ratio (y) is chosen to be one. With this damping ratio, there can be no overshoot of the robot, a condition to be avoided. The undamped natural frequency determines the "stiffness" of the controller. For all controllers, the equations for K_v and K_p are:

$$K = A*2*y*w_n \tag{2.5}$$

$$K = A*w_n \tag{2.6}$$

The industrial controller is usually a proportional and derivative (PD) feedback loop, which is designed to produce critical damping. This is caused by the fact that robot dynamics are non-linear and the PD loop assumes that the dynamics are fixed. Inertia of a robot changes with the position of the robot joints.

Industrial controllers operate adequately due to the nature of geared manipulator dynamics. The gear ratios of the PUMA and other geared manipulators range from 80 to 1 to above 100 to 1 [7]. High gear ratios reduce the significance of load torques produced by link motion as seen by the motor. This allows the controller to ignore coriolis and centrifugal torques and make other simplifying assumptions, such as diagonalizing the inertia matrix, without producing unstable response. This simplification is only valid when disturbances are lower then the controllers disturbance rejection capability, which does limit the speeds and loads that the manipulator can handle.

Actuator Dynamics

An obvious effect of high gear ratios is to increase the importance of accurately describing the motor and gear dynamics in the controller model, because the motor friction terms and actuator inertia are multiplied by the square of the gear ratio. Although its effect is well known in industry, early robot control research ignored the importance of the actuator in modelling the manipulator. Tarn, et al are among the first to document the effect of actuator inertia on the dynamics of a manipulator[11:18] Tarn, et al accurately identified the actuator inertia that should be included in the PUMA motor model[11:17]. Properly describing the actuator is an important component in building an accurate model of the mainpulator.

The static friction of the gear train has also been neglected in past simulators. Several attempts are then made to model the static friction involved in the PUMA actuator.

> "Previous results illustrate high initial position errors that could be the product of a lack of accurate static friction compensation and that nonlinear torque dependent friction compensation is an inadequate form of compensation "[8:152].

Proper simulation of static friction can be accomplished by use of a non-linear velocity dependent switching function [8].

A more complete description of the robot actuator is a standard second order model of the motor. The second order actuator model is given by:

$$T_m = J_{eff}\ddot{\theta} + B_{eff}\dot{\theta} + T_f \qquad (2.7)$$

where:

T_m - motor torque

J_{eff} - actuator inertia

B_{eff} - viscous friction

T_f - static friction

$\dot{\theta},\ddot{\theta}$ - joint velocity,acceleration

The values for J_{eff}, B_{eff}, and T_f are determined experimentally for the particular robot used. In previous research, the viscous friction has always been ignored. By including both viscous and static friction in the model, a more realistic simulation of the PUMA 560 has been achieved.

Some researchers suggest that the motor model itself is unrealistic in making simplifications to second order and must be improved. The equation 2.2 referred to above is the standard L-E dynamics equation used to describe a torque controlled robot.

> "In much of the literature, the actuators providing the drive torques are modeled as pure torque sources, or as first-order lags. This assumption is the Achilles' heel of the class of robot dynamic models represented by Eq. (2.2)" [10:18].

Goor shows that once the motor dynamics are included, the differential equations for each joint of a PUMA becomes a third order equation instead of second order[3:387]. The pole produced by the armature inductance causes the increase to third order in the model. This thesis intends to show that a second order model of the dynamics can accurately represent the actual motor, because the armature inductance is negligible in industrial manipulators.

Dynamics Based Control Laws

Dynamics based controllers seek to remove the current restrictions on industrial manipulators by incorporating known arm dynamics into the control law. The most commonly mentioned dynamics based control law is computed-torque [7]. The computed-torque technique includes feedforward and feedback elements which contain manipulator dynamics information that is ignored by industrial controllers. "The computed-torque technique employs both feedforward and feedback elements to control a robot arm and is a special case of the optimal control law"[7:71]. The inertia matrix is assumed to be modelled accurately, and the error correction from feedback loop is added to the desired acceleration prior to multiplication of the inertia matrix. The computed-torque controller calculates the torque required given the desired acceleration, velocity, and position. The equation for a computed-torque controller is given by:

$$T(t) = D(\theta)[\ddot{\theta}_d + K_v(\dot{\theta}_d - \dot{\theta}) + K_p(\theta_d - \theta)] \; \ldots$$
$$+ h(\dot{\theta},\theta) + g(\theta) + T \qquad (2.8)$$

where:

$\ddot{\theta}_d, \dot{\theta}_d, \theta_d$	- desired acceleration, velocity, and position
$\dot{\theta}, \theta$	- actual velocity and position
K_v, K_p	- constants chosen to attain desired poles
$T(t)$	- torque
$D(\theta)$	- 3x3 inertia matrix
$h(\dot{\theta},\theta)$	- 3x1 coriolis and centrifugal torques
$g(\theta)$	- 3x1 gravity terms
T	- friction torque

If actual and modelled dynamics match exactly, the system dynamics reduce to:

$$D(\theta)[(\ddot{\theta}_d - \ddot{\theta}) + K_v(\dot{\theta}_d - \dot{\theta}) + K_p(\theta_d - \theta)] = 0 \qquad (2.9)$$

or:

$$D(\theta)[\; \ddot{e} + K_v * \dot{e} + K_p * e \;] = 0 \qquad (2.10)$$

where:

$\ddot{e}, \dot{e}, e$ - joint acceleration error, velocity error, and position error

The $D(\theta)$ matrix is always positive definite, which guarantees that its inverse exists. By multiplying both sides by the inverse, a second order unforced differential equation remains. If there are no unmodelled effects, this is an undriven differential equation and, given that K_v and K_p are chosen for stability, the error will be driven asymptotically to zero.

Computed-torque controllers include coupled inertia terms as well as gravity and other terms previously modelled as disturbances. By modelling these terms, the controller has much less disturbances to react to, thus lowering the overall errors involved. "It is found that trajectory tracking errors decreased as more dynamic compensation terms are incorporated"[1:165]. Which terms are modelled is dependant on the desired complexity of the controller to be used. Unmodelled terms are considered disturbances that are rejected by the PD loop.

The two types of dynamics based controllers that are considered for this thesis are the computed-torque and feedforward controllers. The computed-torque controller globally linearizes the system by forcing the poles to maintain critical damping, as long as the basic assumptions are met.

Feedforward controllers locally linearize the system by compensating for changing dynamics terms, leaving a second order, critically damped error expression. The equation for a feedforward controller is:

$$D(\theta)*\bar{\ddot{\theta}}_d + K_v*(\dot{\theta}_d - \dot{\theta}) + K_p*(\theta_d - \theta)] = T_1 \qquad (2.11)$$

The gains are adjusted by multiplying by the minimum self inertia term of the inertia matrix. This locally linearizes the controller at the position where the joint self inertia is minimum. The minimum value is chosen to avoid a undesirable overshoot that can occur if actual self inertia were greater than the assumed self inertia. Assuming that the change in self inertia is small, this equation reduces similarly to the computed-torque.

The link between complexity and sampling rate is an important factor in dynamics based controllers. The more complete the dynamics used in the controller, the more complex the computational algorithm becomes. With complexity comes a slower sampling rate. "Based on our experimental results, we also conclusively establish the importance of high sampling rates as they result in an increased stiffness of the system"[5:169]. Sample rate becomes the key trade off when identifying significant terms to model.

The more complex dynamics based controllers produce smaller errors than a less complex industrial controller when implemented at the same sampling rate and with the same coefficients for the PD loop. But the less complex controller, when implemented at a higher sampling rate, may handle larger PD coefficients, thus outperforming the more complex controller. Khosla maintains that Kv and Kp are a function of the sampling rate [5:171]. Although his research is conducted on a direct-drive manipulator, the conclusions can be extended to geared manipulators. At some point, increased sampling rate will not provide an increase in the stiffness of the system. The point at which the benefits level off is not known because the manipulator model is not known well enough to identify the high frequency component that is being excited at the lower sampling rates. Khosla experimentally determined the maximum K_v possible, at the particular sampling rate chosen, and said that this is limited by the unmodelled high frequency dynamics of the manipulator [5].

The trade off for the increase in the accuracy of the dynamics based controller is in increased complexity of the controller. Previous efforts have been made to bypass the computational burden problem. Multiple processors working in parallel has been suggested by Lee [9]. This would divide the burden and allow increased sampling rates on the feedback loop. Multiple processors would allow computations to be done rapidly, also allowing increased sampling rates in the feedback loop. Another tact could be to divide the feedback and feedforward portions of a controller. The sampling rate of the feedback loop could remain high, while the feedforward portion is updated at much slower rates due to the lower rates of change of this portion. Taken to an extreme, this approach leads to a digital feedforward portion and an analog feedback loop.

Summary

While many researchers have created manipulator simulations, the validity of the modelling assumptions are somewhat suspect. A better understanding of the manipulator being evaluated must be gained, if a validated simulation is to be produced. The advantages of analog simulation, coupled with the ability to test hybrid controllers, compels the researcher to utilize the SIMSTAR for implementing the simulation.

Chapter 3

Simulator Development

Introduction

The main thrust of this thesis effort is the development of an analog simulation package capable of testing digital, analog, and hybrid controllers for robot manipulators. Once developed, this simulation environment will allow real-time testing of control algorithms and real-time simulation of man-in-the-loop systems.

An accurate model of a robot arm is the key ingredient to an accurate simulation package. The PUMA 560 is being used as the case study for this thesis because it is representative of the class of industrial robots appropriate for Air Force research and experimental evaluation of the PUMA exists for validation of the simulation. An accurate model is obtained by building upon previous modelling work, and by experimentally obtained motor model parameters. Once an accurate model is obtained, it is programmed into the analog section of the SIMSTAR hybrid computer. The model is then be validated by exercising several control algorithms over a known trajectory and comparing the position error data with actual data obtained from the PUMA over the same trajectory.

This chapter is divided into two sections to explain the method used to produce a valid simulation package. The first section will cover how the proper model of the PUMA 560 is determined. The second section will explain how the model is programmed on the SIMSTAR hybrid computer.

Modelling

Accurate modelling of the robot arm is the most significant contributer to a realistic simulation environment. The modelling of the arm is divided into modelling the dynamics of the links, modelling the dynamics of the motor and drive unit, including gearing, and identifying/reducing the significant terms in the model.

The dynamics of the links of a robotic manipulator are often described by the Lagrange-Euler formulation, because this set of equations is easily understood and link coupling is clearly shown. Other formulations, such as Neuton-Euler, are also used and can be much less computationally intense, but these formulations do not give the control engineer the insight into the dynamic coupling that Lagrange-Euler formulation provides. The overall Lagrange-Euler robot dynamics of the first three joints of the PUMA 560 can be described in terms of 3 non-linear, coupled, differential equations.

$$nT_m = n^2 J_m \ddot{\theta} + n^2 B_m \dot{\theta} + T_f + T_l \qquad (3.1)$$

where:

n - 3x3 diagonal matrix of gear ratio

T_m - 3x1 vector of motor torque

J_m - 3x1 vector of actuator inertia

B_m - 3x1 vector of actuator viscous friction

T_f - 3x1 vector of static friction torque

T_l - 3x1 vector of load torque

These terms can be separated into contributions from the links of the robot and from the actuator/motor of each joint. The terms that are dependant on θ are link terms, while the static friction and actuator terms are contributed by the actuator/motor.

Link Dynamics. The link dynamics of a robot arm are described by highly coupled, nonlinear differential equations. The link dynamics can be described by the following equation:

$$T_l = D(\theta)\ddot{\theta} + h(\theta,\dot{\theta}) + g(\theta) \tag{3.2}$$

where:

T_l	- 3x1 vector of joint link torques
$\theta,\dot{\theta},\ddot{\theta}$	- 3x1 vectors of joint position, velocity, and acceleration
$D(\theta)$	- 3x3 inertia matrix
$h(\theta,\dot{\theta})$	- 3x1 vector of coriolis and centrifugal terms
$g(\theta)$	- 3x1 vector of gravity terms

Each term in the $D(\theta)$ matrix represents the inertia effect of each link's acceleration. The off diagonal terms are the inertial coupling terms, i.e. the effect of other joint's acceleration on a joint. These terms are dependent on the current position of each joint, as the inertia of the robot is different for every arm configuration. The PUMA 560 equations for $D(\theta)$ are taken from Tarn [11], and are:

$$D11 = 2.4975 + 2.1007*\cos(\theta2)2 + 0.5323*\sin(\theta2+\theta3)2 + - 0.033*\cos(\theta2)*\cos(\theta2+\theta3) + 0.9161*\cos(\theta2)*\sin(\theta2+\theta3) \tag{3.3}$$

$$D22 = 5.419 + 0.9161*\sin(\theta3) - 0.0331*\cos(\theta3) \tag{3.4}$$

$$D12 = 2.4492*\sin(\theta2) + D13 \tag{3.5}$$

$$D13 = -0.007*\sin(\theta2+\theta3) - 0.1596*\cos(\theta2+\theta3) \tag{3.6}$$

$$D23 = 0.5468 + 0.4581*\sin(\theta 3) - 0.0165*\cos(\theta 3) \quad (3.7)$$

$$D33 = 1.1295 \quad (3.8)$$

The D(ij) term is in the i^{th} row and j^{th} column of the matrix. The inertia matrix is symetric, so the D(ij) term is the same as the D(ji) term.

The $h(\theta,\dot{\theta})$ term is derived from a 3x3x3 tenser dependent both on joint positions, and velocities. This term contributes significantly to the computational intensity of the overall dynamics. At high angular velocities this term may supply a significant portion of the torque created by the robot; however, for velocities currently used in industrial applications, the $h(\theta,\dot{\theta})$ term provides a insignificant portion of the overall geared manipulator torque. Its significance is reduced because the high gear ratios of the industrial manipulator reduce the link torque as seen by the motor. Because of its insignificance at typical speeds and its computational intensity, it is ignored in all geared manipulator controllers used in this research.

The gravity term, $g(\theta)$, is dependent on the position of each of the joints. Joint one of the PUMA 560 is not affected by gravity because the torque applied by that motor is perpendicular to the gravity vector, assuming the robot is mounted on the floor. The equations for the additive gravity torque for joints two and three have been identified by Tarn [11]. The equations describing the torque are:

$$g(\theta 2) = -52.106 * \cos(2*\theta 2) +1.0972 * \sin(2*\theta 2) +g(\theta 3) \quad (3.9)$$

$$g(\theta 3) = 0.3761 * \cos(\theta 2 + \theta 3) - 10.4068 * \sin(\theta 2 + \theta 3) \quad (3.10)$$

As can be seen from the first coefficient in Equation 3.8, the gravity component can provide a significant portion of the torque on joints two and three. Proper description of the gravity torque is very important to create an accurate model.

Motor Model. Previous robot simulations have made simplifications in the motor model that seriously compromised the accuracy of the overall robot model [8]. Static and viscous friction coefficients are necessary to accurately model the robot. The inertia term from the motor also affects the overall model accuracy. The high gear ratios of the industrial manipulator is the reason an accurate motor model is of such importance.

As seen by the motor, the torque created by link dynamics and gravity are divided by the high gear ratios. This reduces the importance of the link torque, while increasing the importance of the motor torque. The torque on the motor side of the gears is related to the link torque by the equation:

$$T_m = 1/n(T_l) \tag{3.11}$$

The equation relating current input to torque output for the motor is:

$$nJ_m\ddot{\theta} + nB_m\dot{\theta} + 1/nT_l = K_T I_m \tag{3.12}$$

Assuming that the link torque can be compensated for by using knowledge of link dynamics, the transfer function of the motor becomes:

$$(J_{eff} + D(ii))\ddot{\theta} + B_{eff}\dot{\theta} = T_{eff} \tag{3.13}$$

Where:

$$J_{eff} = n^2 J_m$$

$$B_{eff} = n^2 B_m$$

$$T_{eff} = nT_m - T_f$$

In the s-domain, the transfer function of joint velocity wrt torque becomes:

$$\dot{\theta}/T_{eff} = 1/[J_{eff}S + B_{eff}] \qquad (3.14)$$

Static friction is modelled by Leahy using a nonlinear, velocity dependent, switching function [7]. The static friction term is determined by applying increasing torque to the arm to determine the amount of torque required to just overcome the stiction.

Once the terms for motor dynamics are determined, the overall model for the positioning joints can be attained by combining the link dynamics determined by Tarn [11], the static friction used by Leahy[7] and the motor dynamics experimentally determined. The mathematical model is:

$$T(t) = D(\theta)*\ddot{\theta} + h(\theta,\dot{\theta}) + G(\theta) + T_f + B_{eff} * \dot{\theta} \qquad (3.15)$$

The Jeff terms from the motor model are added to the self inertia term in the D(θ) matrix.

Step Test. Link dynamics for the PUMA have been well described by previous researchers [11]. However, motor dynamics are mostly neglected in previous simulations. Static friction is added, in recent work, as well as actuator inertia [8]. It is necessary to identify the viscous friction coefficient of the motor to create an accurate simulation environment.

To identify the motor dynamics, it is necessary for the motor dynamics to be observable. By minimizing the link dynamics, and extracting the known terms, motor dynamics are the remaining effects on the joint position and velocity. To determine the motor dynamics experimentally, step tests are run using a known torque input. Each joint is tested separately while the other joints are held stationary. This eliminates most of the link dynamics terms, basically leaving the self inertia term to be compensated for in the drive torque. The remaining dynamics are due to the motor. The PUMA 560 orientation for the test is chosen to minimize the effect of gravity across the trajectory. Velocity data is recorded, and compared to response data generated by an ideal motor model. The ideal model data is fit to the actual data, thus identifying the correct viscous friction terms to be used. This data is curve fit to mathematically produced data on MATRIXx.

The input for the step tests are based on counts. Counts are proportional to current by the equation:

$$C = K_C * I \tag{3.16}$$

where:

C - counts

K_C - constant

I - motor current

Several levels of counts are run in the step tests. Data is taken and cataloged for each level of counts.

$$\dot{\theta}/((C^{-} - T_f)/nK_C) = 1/[J_{eff}S + B_{eff}] \tag{3.17}$$

where:

$J_{eff} = n^2 J_m$

$B_{eff} = n^2 B_m$

C - counts

T_f - static friction torque

The static friction is not compensated for in the controller. Instead, the ideal step responses generated mathematically are compensated by subtracting known static friction torque from the step function.

The velocity data collected from the step tests are placed in MATRIXx for comparison to ideal model responses. MATRIXx is capable of generating time response data from an ideal model of the motor. The ideal model is generated using equation 3.2, and the ideal response is compared to the actual PUMA data. The coefficients used in the ideal model are adjusted to fit the ideal response to the PUMA response. Once the best fit is found, using engineering judgement, the coefficients for viscous friction had been found.

Plots for each joint are shown in Figures 3.1 - 3.6 comparing actual and ideal responses for different magnitude step inputs. Response to different step inputs can be found in Appendix E.

In the actual velocity response, there is a clipping function inherent in the PUMA. This does not effect the validity of the ideal model, because joint velocity is restricted below the clipped value in trajectory generators. The velocity data is used for comparison to the ideal model. Each experimental response is slightly different, because the actual model is not exactly described by the motor model alone.

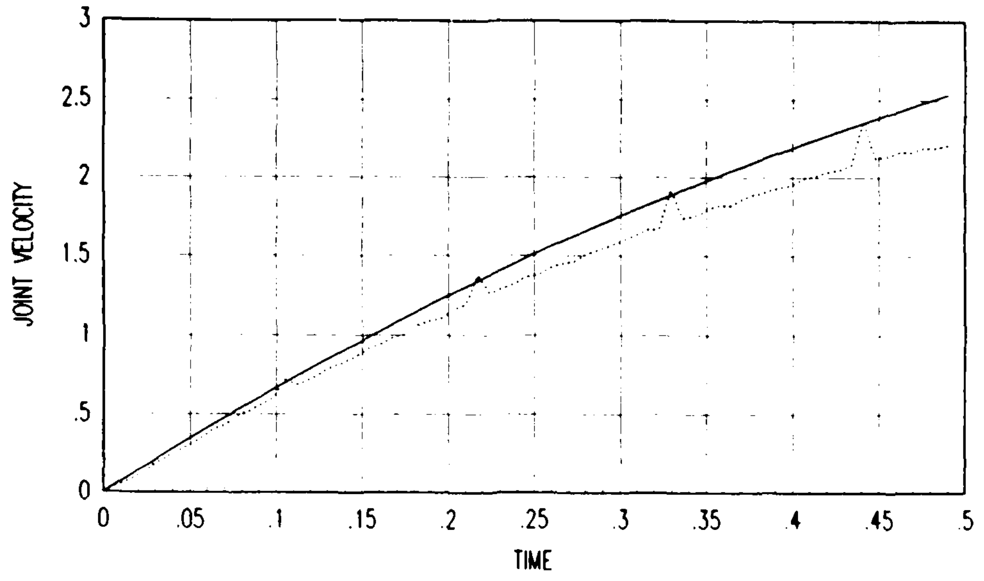


Figure 3.1. Joint One Step Response, Counts = 500

Actual Joint Response ....
Ideal Step Response ________

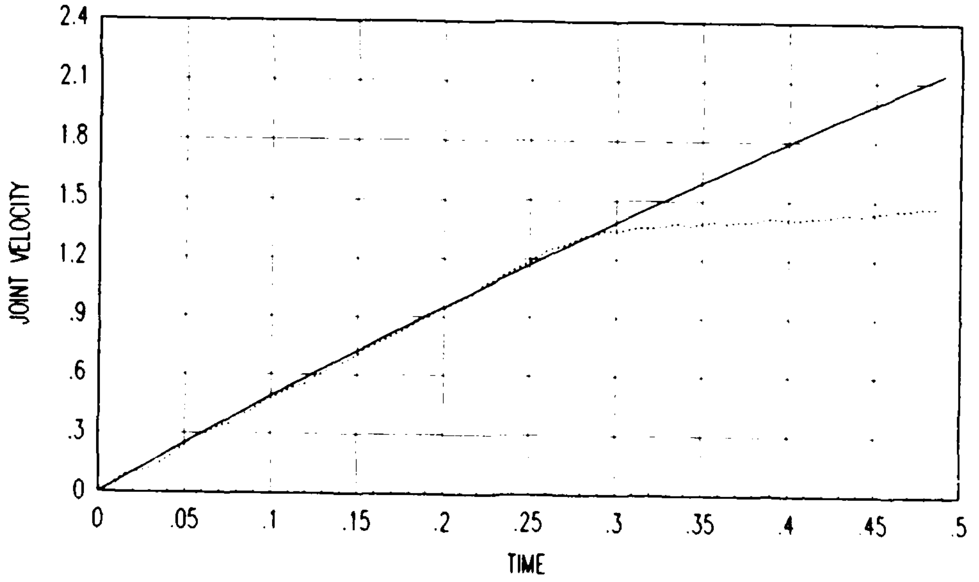


Figure 3.2. Joint Two Step Response, Counts = 400

Actual Joint Response ....
Ideal Step Response ________

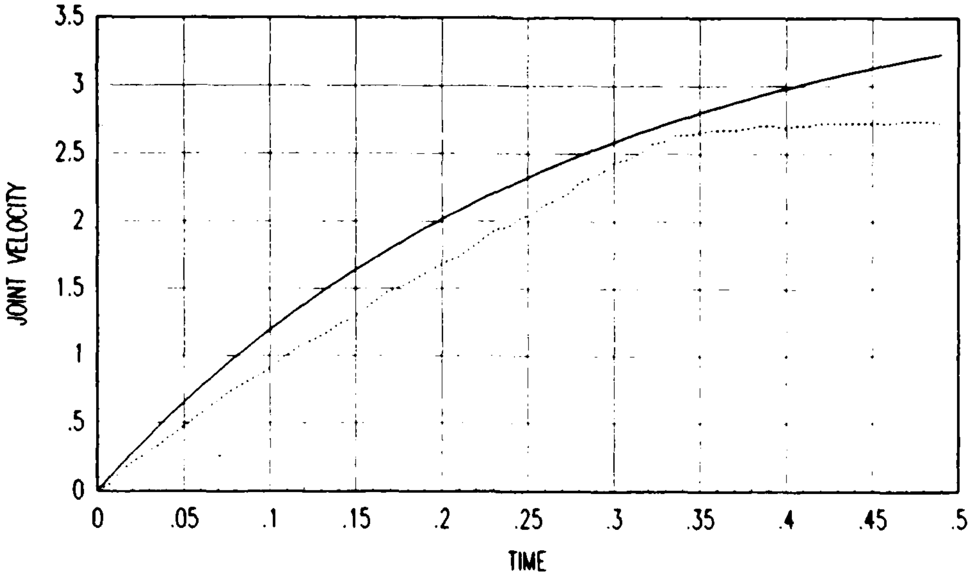


Figure 3.3. Joint Three Step Response, Counts = 400

Actual Joint Response ....
Ideal Step Response ________

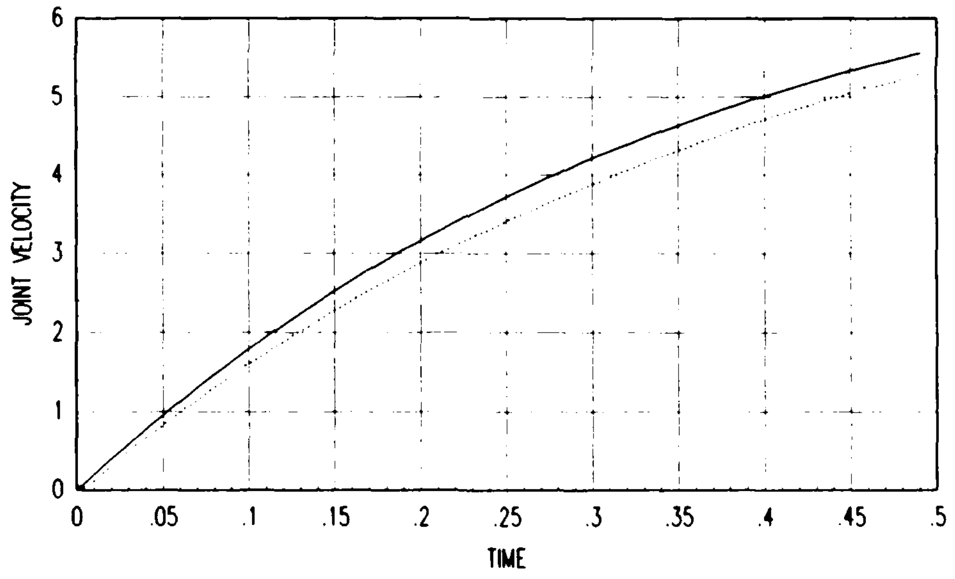


Figure 3.4. Joint Four Step Response, Counts = 400

Actual Joint Response ....
Ideal Step Response ________

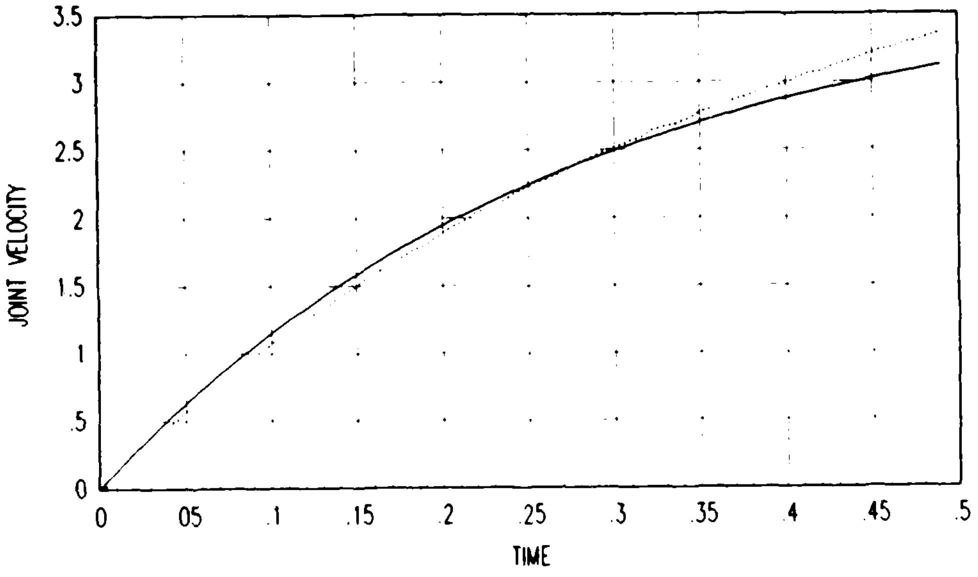


Figure 3.5. Joint Five Step Response, Counts = 300

Actual Joint Response ....
Ideal Step Response ________

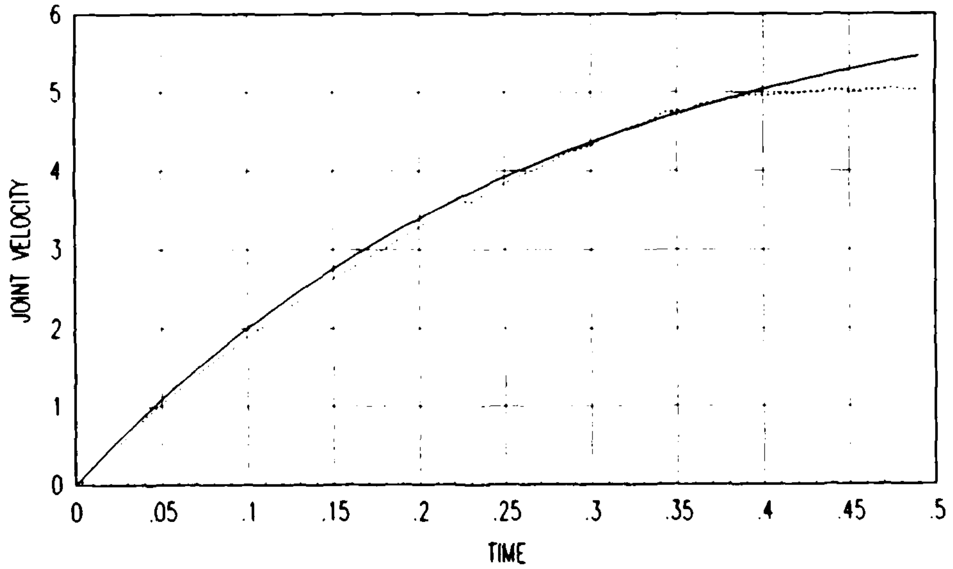


Figure 3.6. Joint Three Step Response, Counts = 500

Actual Joint Response ....
Ideal Step Response ________

Viscous friction coefficients are determined by assuming a known static friction value and by assuming a known J_{eff}. This leaves the B_{eff} as the only unknown in the equation. Table 3.1 shows the values used for each known, and the values found for the viscous friction.

Table 3.1

Motor Dynamics Coefficients

Joint	T (counts)	J_{eff}	B_{eff}
One	125	2.54	3.5
Two	75	5.2	3.5
Three	89.6	1.08	3.5
Four	90.5	0.18	0.48
Five	90.8	0.15	0.55
Six	90.2	0.18	0.65

The values for B_{eff} are used in modelling of the PUMA in the SIMSTAR. The viscous friction coefficients are multiplied by the joint velocity to generate the damping torque created by viscous friction.

Simulator

The SIMSTAR hybrid computer at AFIT has several unique features. The SIMSTAR is capable of integrated digital and analog computing, with the same time reference, and with internal scaling of the analog variables. This allows a user to implement an analog model while retaining the ability to test digital controllers.

The digital computing is done in the Digital Arithmetic Processor(DAP), a Gould 32-27. It is connected to the Parallel Simulation Processor(PSP) through the Parallel Logic Unit(PLU). The PLU is tasked with connecting the PSP, analog section, in the correct configuration to simulate the model programmed in the DAP. It is also tasked with handling digital-to-analog and analog-to digital transfers. The PLU simplifies the task of implementing an analog simulation, because the user no longer has to patch the analog section by hand.

The SIMSTAR programming is broken up into several different languages. The basic program structure in the SIMSTAR is standard in all programs. The following is the basic programming structure.

```
*PSP=1,0,ERR=ALL
*TITLE
Title of Program
*INPUT
PROGRAM
     INITIAL
     '@BETA(1)'
     END $'INITIAL'
          DYNAMIC
               DERIVATIVE
                    '@PARALLEL'
                    TERMT(TIME .GT. RUNTIM )
                    '@ENDPARALLEL'
               END $ 'OF DERIVATIVE'
          END $ 'OF DYNAMIC'
     TERMINAL
     END $ 'OF TERMINAL'
END $ 'OF PROGRAM'
*TRANSLATE
*OUTPUT
*END
```

The initial and derivative sections are programmed in DTRAN, a language developed by the makers of the SIMSTAR. DTRAN allows the user to program the digital section using Applied Continuous System Language (ACSL) constructs. DTRAN is an unusual language because it is not sequential. The DAP processes the statements according to how the compiler decides to arrange them. This can be avoided by declaring an implicit region, where the programmer declares which statements are processed first. The implicit region is programmed as follows:

```
PROCEDURAL( LHS variables = RHS variables)
'@IMPL (    variables)
statements
'@END IMPL'
END
```

where:
    LHS - Left Hand Side
    RHS - Right Hand Side

The PROGRAM region contains the entire program to be run, including both analog and digital portions. Variables can be declared in any region within the PROGRAM region. Interrupt declarations are usually made just prior to the INITIAL region, in the PROGRAM region. The INITIAL region is executed once, when the routine is run. It is used to initialize variables.

The DYNAMIC region is executed continuously once the program is started, until the program is timed out. The DYNAMIC region contains both the analog and digital portions. The DYNAMIC region is broken up into the DERIVATIVE region, which contains the digital routine, and the PARALLEL region, which contains the analog routine. The only programming outside this region are FORTRAN 77 subroutines, which are placed after all of the programming regions, and called from within the PROGRAM region.

The DERIVATIVE region contains the digital controller used in this thesis. For each different controller, a different program is generated. The feedforward controller that includes coupling terms is in the program S1.FFFG, which can be found in Appendix A. The analog model is the same for each program, while the DERIVATIVE region contains the controller. It also contains equations that define the A/D conversions used in the program. Basically, model generated velocities and positions are sent to the DERIVATIVE region, where they are used in the controller to generate a digital torque value. This torque is sent through a D/A conversion to drive the analog model.

The desired trajectory is loaded into arrays in the initial region, and is used by the controller in generating the torque. The array position is referenced by a digital pointer that is based on the sample rate of the program. The sample rate in SIMSTAR programs is controlled by a variable called CINT. In this thesis CINT is set to 14 ms so that the data collected could be compared to actual PUMA data generated at 14 ms.

The static friction compensation generated for the controller is calculated in a FORTRAN subroutine. Velocity and torque are used to determine static friction direction. This torque is added to the controller torque before it is sent to the analog model.

Gravity compensation is generated by two equations in DTRAN, based on actual joint position. This torque is also added to the torque generated by the controller. In the feedforward controllers, viscous friction compensation is also generated.

The program that contains the feedforward controller with diagonal inertia terms is called S1.FFDG. The program can be found in Appendix A. It contains the same compensation terms as the S1.FFFG controller, but it does not couple the controllers with the off diagonal inertia terms.

The program that contains the PD controller is called S1.PDG.

It contains gravity and static friction compensation, but does not feedforward any information about desired acceleration. This program can also be found in Appendix A.

The PARALLEL region contains the analog model used in this thesis. It completes all of the computations continuously through use of analog summers, multipliers, comparators, sin/cos function generators, and other analog components.

Each variable that is based on the function of θ (Q in the programs) requires sin and cos generators to calculate the terms. To minimize the number of sin/cos generators, the first section in the PARALLEL region calculates all sin and cos terms. Also in this section are multiplication terms that are used repeatedly in the model. They are given a variable name to conserve the number of multipliers used.

The next section of the PARALLEL region contains the equations that control the D/A conversions. They are made up of torque transfers from the controller.

The static friction term is a nonlinear function of velocity, and requires several special switching function generators to calcuate the static friction torque. If the absolute joint velocity is greater than 0.01 rads/sec, the static friction constant takes the sign of the velocity. If the velocity is less than 0.01 rad/sec, the static friction constant takes the sign of the torque term.

Viscous friction compensation torque is calculated by multiplying B   by the velocity of the joint. This torque value is then added to the controller torque, static friction torque, and the gravity compensation. This torque value is the driving input to the differential equations that simulate the robot arm.

The model equations are divided into algebraic equations that consider position, velocity, and acceleration as separate variables in the equation, and integrations that link the position, velocity, and acceleration of each joint. Because robot dynamics are coupled, it is necessary to put this section into an IMPLICIT region. This programming structure explains to the compiler how to link up the variables internally in the analog portion.

Finally, the PARALLEL region contains the time function which is generated by a integrator inside the analog section of the SIMSTAR. The end of run interrupt is based on this time function exceeding the range of runtim.

The D/A conversion in the SIMSTAR is accomplished by a zero order hold that takes the digital value and holds it constant over the entire sample period. The sample period used for this simulation is 14 msec. The A/D conversion has a transfer time of approximately 50 microseconds.

Summary

To properly complete this thesis, it is necessary to work step by step through development of the model, implementation on the SIMSTAR, and validation of the model. The model is a combination of previously developed model and experimental evaluation. Implementation on the SIMSTAR involved streamlining the computations in the analog portion, and solving SIMSTAR related limitations.

Chapter Four

Experimental Validation

Validation of the simulation was necessary before it is used to test and compare different controller designs. Verification is performed on the simulation by comparing SIMSTAR generated joint errors to PUMA 560 generated joint errors. If the differences in these error profiles are to be confined to simulation errors, it is necessary to exercise the simulation and PUMA 560 using the same trajectory and controller. By holding trajectories and controllers constant, the simulation is subjected to the same link torque profile to which the acual PUMA is subjected.

Data Reduction. Data for this thesis is collected in the form of joint position and angular velocity arrays. The array data is referenced to time with each array position being 14 ms further in time. MATRIXx is used extensively to process and display data. It is also used to generate ideal response data to compare to experimentally generated data.

The SIMSTAR analog model of the PUMA 560's positioning joints is exercised, using a known trajectory, by implementing three different controllers. Trajectory error data, in the form of position error matrices, is collected on the SIMSTAR . This data is also transferred to the Instrumentation Sciences Laboratory (ISL) VAX 11-780 for analysis and comparison to error data collected from AFIT's PUMA 560.

The software package, MATRIXx, is used to interpret the data, and to generate ideal step response data for identification of proper motor model coefficients. The communication protocol, Kermit, is used to transfer data between computers. Programs used to configure the data files for use in MATRIXx can be found in Appendix B.

Error Profiles

Once the analog model had been completed, it is necessary to exercise the model using a known trajectory and known controllers to determine the extent of the accuracy of the model. The trajectory is determined based on available trajectory generation, and available data taken from the AFIT PUMA 560 over that trajectory. The initial trajectory is generated in the SIMSTAR using a routine called S.TRAJEC(see Appendix B). This trajectory uses a symmetric velocity with a peak equal to the maximum velocity of each joint. Inital conditions are chosen based on apriori knowledge of error profiles that are generated, by the same trajectory and initial conditions, on the PUMA 560. Joints one and three are moved through 90 degrees($\pi/2$ rads) while joint two is restricted to 45 degrees($\pi/4$ rads). The restriction on joint two is caused by a velocity restriction on the joint. The model is run through the entire trajectory in 1.5 seconds.

A problem with this trajectory is that it directed the joints to change acceleration from positive to negative in such a short period of time that it violated the actuators' jerk constraints. This problem can be avoided in the simulation by increasing the scaling of the analog variables, but there would be no actual error profiles with which to compare.

The original trajectory is used for initial debugging because it is easy to generate on the SIMSTAR. This trajectory is eventually replaced by a trajectory that avoided most of the PUMA's jerk constraint. This trajectory had the same initial position and end points, but is generated external to the SIMSTAR. This trajectory, identical to the one used for experimental evaluation, is generated by connecting cubic splines, not by one mathematical equation, so it couldn't be programmed on the SIMSTAR. The data plots used for this trajectory can be found in Appendix B.

To validate the simulation, the model is subjected to the three different controllers over a desired trajectory. Joint position error data is collected at every sample period. The AFIT PUMA 560 is subjected to the same controllers, over the same trajectories, with joint position error data collected at the same rates. These error profiles are then compared to verify that the simulation did react like the actual PUMA. Simulation error profiles are expected to give trend information, as opposed to exact errors. Because only trend information can be expected from simulation, there is no substitution for actual experimental evaluation for final testing of an algorithm.

Each controller adds complexity by including increased dynamics-based feedforward terms. In this way the model will see the full range of complexity in controllers. Also, any mismodelling may be isolated by use of different terms in each controller. The trajectory is chosen to exercise the model with high velocities while avoiding constraints on the PUMA 560.

The first controller is a basic PD controller with static friction and gravity feedforward compensation. The equation used for each joint is:

$$T(t) = K *(\dot{\theta}_d - \dot{\theta}) + K (\theta_d - \theta) + G(\theta) + T_f \qquad (4.1)$$

where:

T(t) - controller torque

Each joint controller coefficients are found based on the minimun self inertia term, and chosen to place the poles at (s+15) in the s-plane. This controller treats each joint independently. Also a factor in the choice of poles is the necessity to compare model generated error trajectories to PUMA 560 generated error trajectories. Table 4.1 shows the loop coefficients used in this controller.

Table 4.1

PD Loop Coefficients

Joint	K_v	K_p
One	70.6	563.4
Two	152.9	1172
Three	25.0	215

The initial conditions used for all of the tests are chosen to cause the gravity torque to contribute greatly to the overall torque. Table 4.2 shows the initial conditions.

Table 4.2

Initial Conditions

Joint	I. C. 0	I. C. 1	I. C. 2
One	0.0	0.0	90.0
Two	-90.0	-135.0	0.0
Three	90.0	135.0	0.0

These initial conditions are added to the base trajectory to produce the actual trajectory used in the simulation, because the base trajectories assume all joints begin at zero degrees.

The joint position error profiles generated by the simulation deviated significantly from the expected errors. Figures 4.1-4.3 show simulation vs. actual position error profiles. Initial condition one data is shown, other initial condition data can be found in Appendix D.

Figure 4.1 shows joint one position error vs. time. The error in the simulation is smaller than the actual error, but it shows the correct direction of the error. The final error does not rise back above zero, but the error trend is similar to the actual arm.

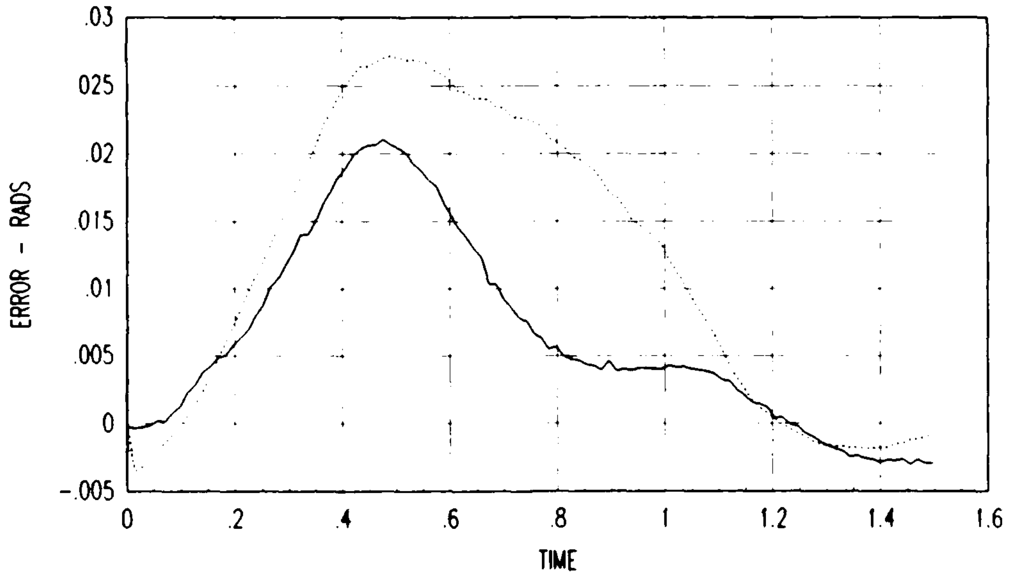


Figure 4.1. Joint One PD Error Profile (I.C. 1)

Simulation ______
Actual ......

Joint Two error data is shown in Figure 4.2. The simulation errors are smaller than the actual errors, but does show the error direction accurately. Final errors are much smaller in the simulation.

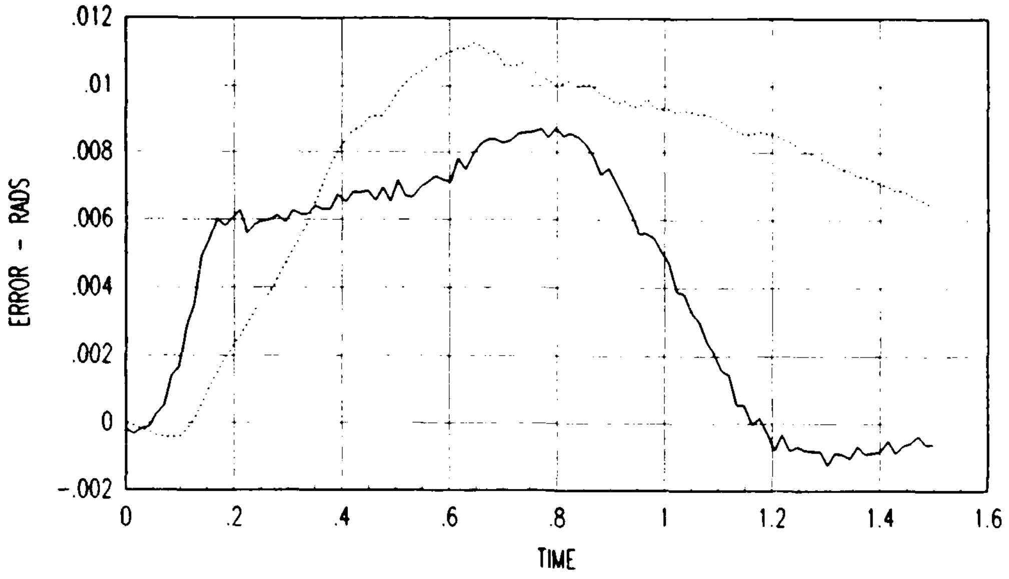


Figure 4.2. Joint Two PD Error Profile (I.C. 1)

Simulation ______
Actual ......

Joint three data is shown in Figure 4.3. The simulation data accurately portrays the actual error profile. This error is a typical second order PD response.

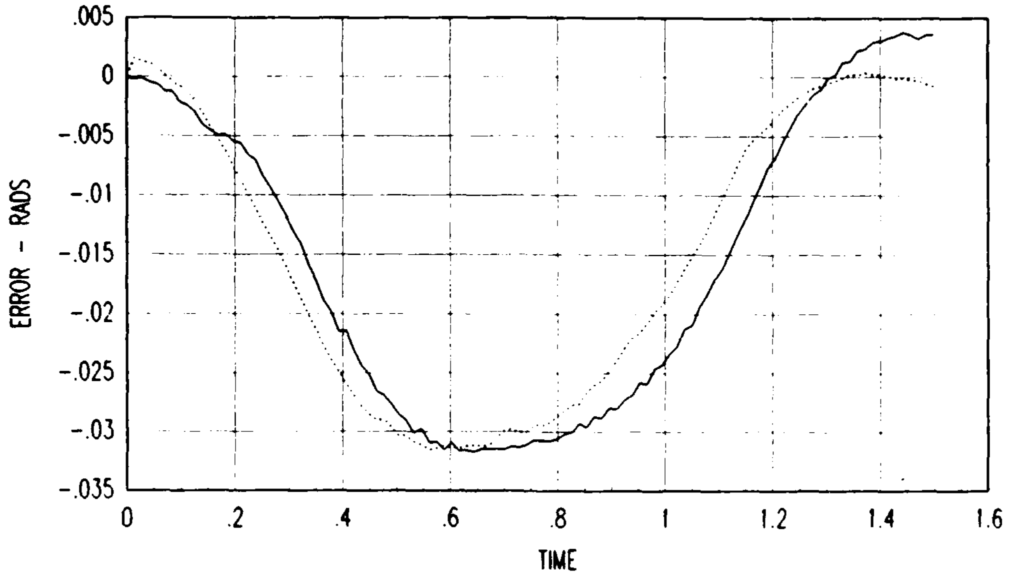


Figure 4.3. Joint Three PD Error Profile (I.C. 1)

Simulation ______
Actual ......

Overall response of the simulation to the PD controller yielded the desired trend information. Joint two exhibited an end point mismodelling of approximately 0.007 radians (0.4 degrees). While this is not a serious mismodelling, as the model did track the trajectory as expected, the inability to trust simulated end point accuracy needs to be eliminated. Joints one and three gave good representative error trend information in terms of mid course magnitudes and end point errors.

The second controller used is a feedforward controller with independent controllers for each joint. This means that the non-diagonal terms of the D(θ) matrix are assumed zero. The coefficients for the PD portion of the controller are determined such that the poles cannot become overdamped. The controller is the same as the PD controller except that the diagonal inertial terms from the D(θ) matrix are multiplied by the desired acceleration term for the joint. The poles for this controller are the same as the PD, but the velocity error gain is different to compensate for the viscous friction modelling. The viscous friction term adds a velocity error term similar to the velocity feedback term. The loop coefficients for this controller are given in Table 4.3.

Table 4.3

Feedforward/Diagonal Coefficients

Joint	K_v	K_p
One	75.12	563.4
Two	156.4	1172
Three	28.66	215

The simulation response to the feedforward controller is representative of actual PUMA responses. See Figures 4.4 - 4.6 for error profile comparison.

Joint One data is given in Figure 4.4. Joint One response shows accurate representation of the error profiles. As expected, the simulation leads and then lags for each joint when the feedforward diagonal controller is used.

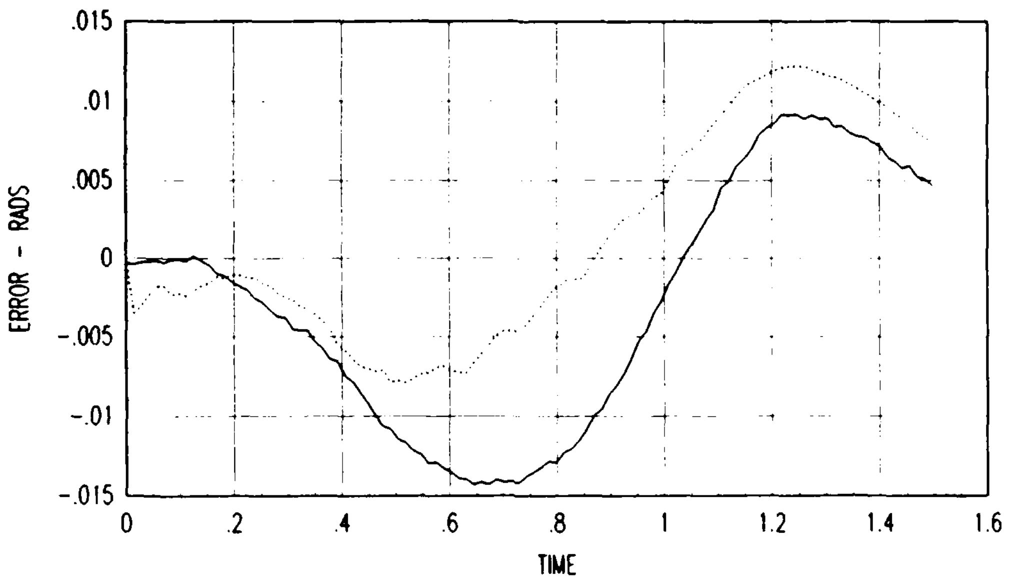


Figure 4.4. Joint One Feedforward/Diagonal Error Profile (I.C.1)

Simulation ______
Actual ......

Joint Two errors are shown in Figure 4.5. Joint Two response errors give good error trend data. The lag in simulation profile vs PUMA is an interesting phenomena that shows up using the feedforward diagonal controller. However, this does not effect the validity of the trend information provided.

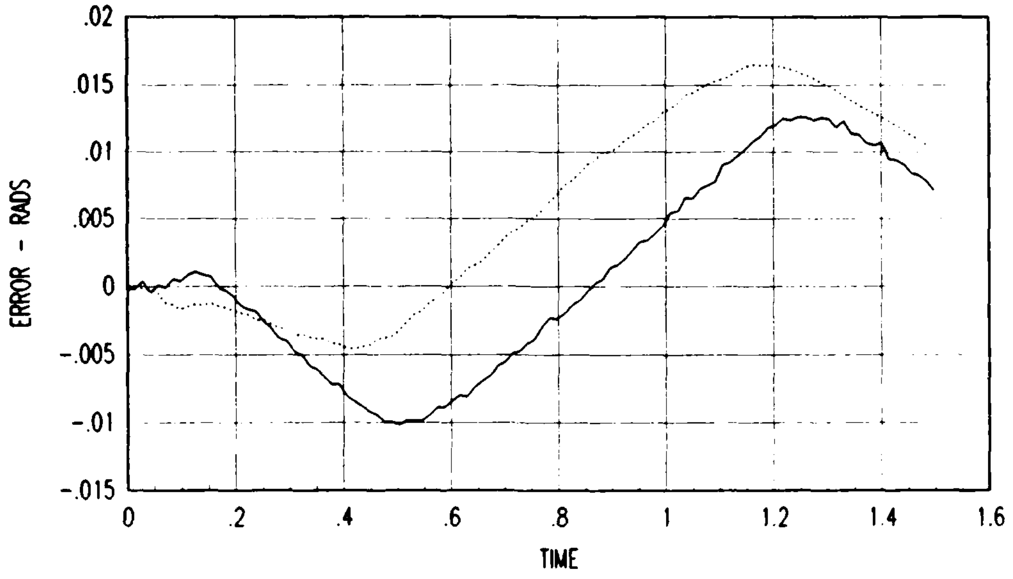


Figure 4.5. Joint Two Feedforward/Diagonal Error Profile (I.C.1)

Simulation ______
Actual ......

Joint Three data is given in Figure 4.6. Joint three's simulation is the most accurate of the three joints.

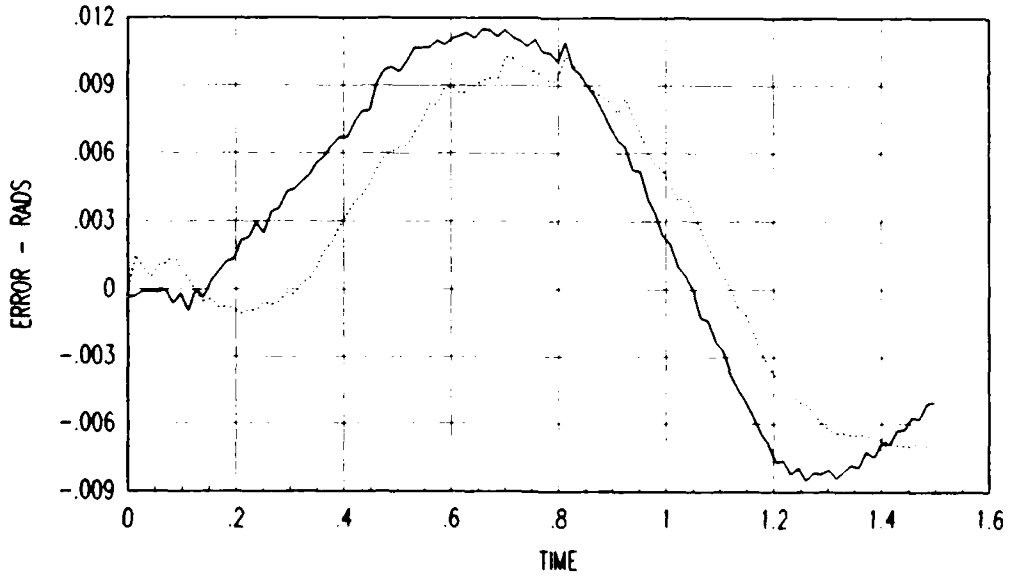


Figure 4.6. Jt Three Feedforward/Diagonal Error Profile (I.C.1)

Simulation ______
Actual ......

The simulation provided accurate mid course and final position errors for each joint. It is important to note that end point errors are on the order of 0.002 radians (0.11 degrees).

The third controller is also a feedforward controller, with the off diagonal terms from the D(θ) matrix included. This provides the coupling between the joints. Otherwise, it is the same controller as the previous feedforward controller. The poles of the controller are given in Table 4.4.

Table 4.4

Feedforward/Full Coefficients

Joint	K_v	K_p
One	75.12	563.4
Two	156.4	1172
Three	28.66	215

Error profiles again suggest that joint three is the most accurately modelled of the three joints. Figures 4.7 - 4.9 show the error profiles generated by the feedforward/full controller.

Joint One data is displayed in Figure 4.7. Joint One mid course errors are overestimated by the simulation. Notice that as time increases, the mismodelling error increases almost constantly.

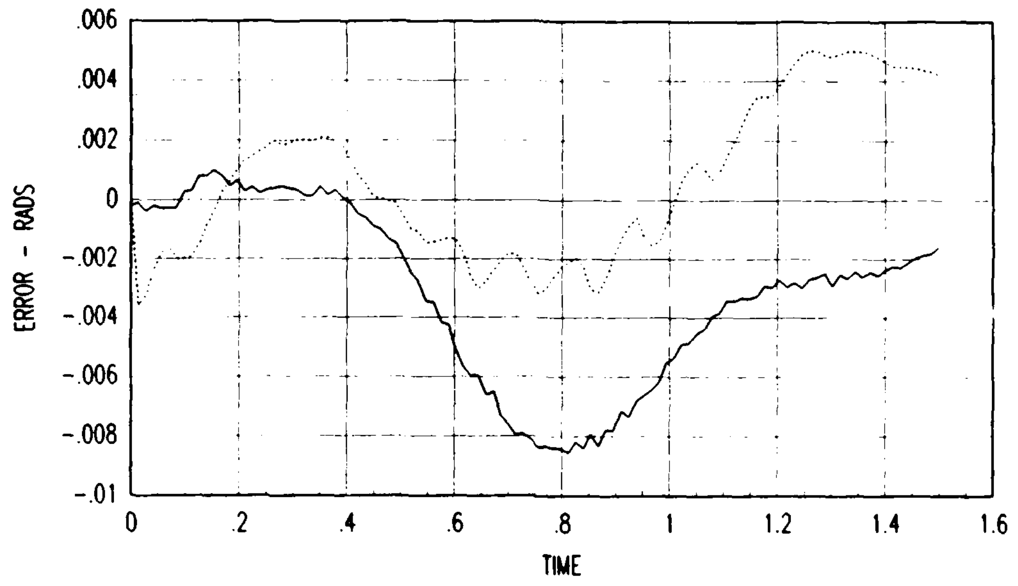


Figure 4.7. Joint One Feedforward/Full Error Profile (I.C. 1)

Simulation ______
Actual ......

Joint Two data is given in Figure 4.8. The simulation underestimated the actual errors, but did show proper error direction. It is again apparent that a mismodelling is increasing the modelling error constantly through the trajectory. This seems to be a bias error that is introduced into the dynamics of the model.

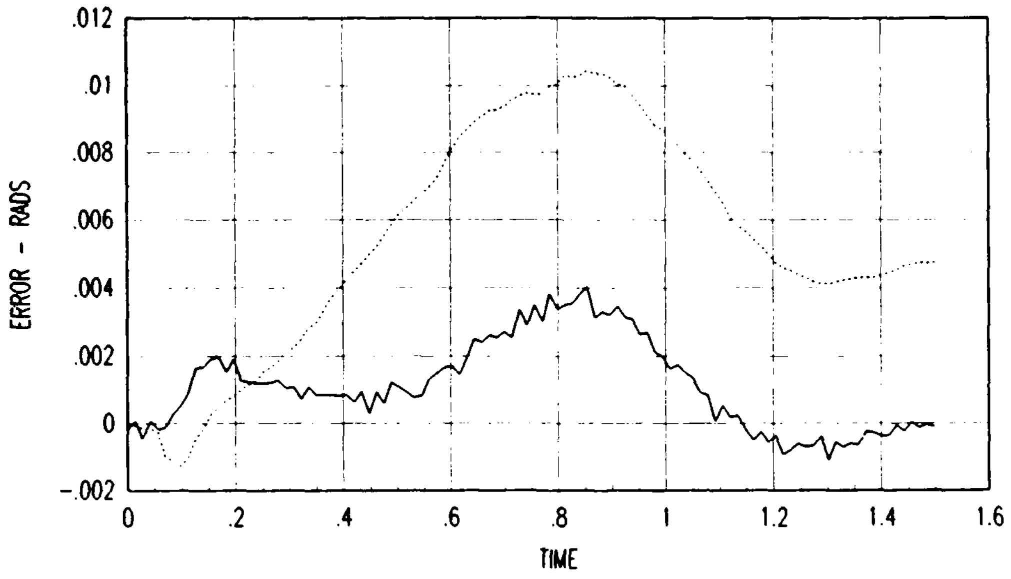


Figure 4.8. Joint Two Feedforward/Full Error Profile

Simulation ______
Actual ......

Joint Three data is given in Figure 4.9. Joint three simulation results are representative of the actual errors. In this case, the mismodelling does not follow the pattern of the previous two joints. It seems to be more an underestimated error, than a bias.

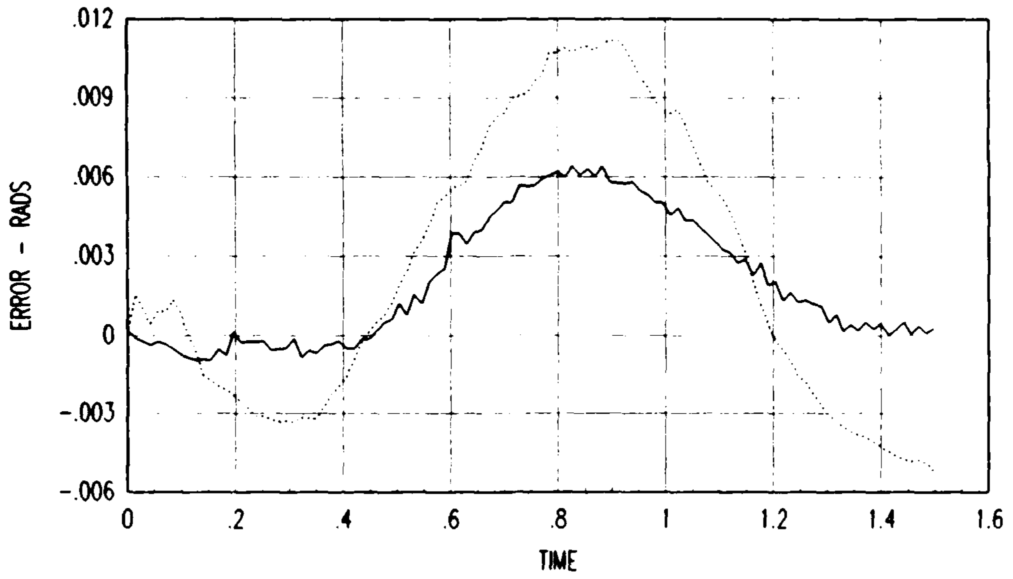


Figure 4.9. Joint Three Feedforward/Full Error Profile (I.C. 1)

Simulation ______
Actual ......

The bias error that occurs in joints one and two should be eliminated. However, the magnitude of the mismodelling is small, 0.006 radians. The model response to the trajectory input is proper in that it followed the trajectory to the final end position. This would suggest that the mismodelling is minor in the sense of overall response, but significant in the sense of comparing errors produced to actual PUMA response.

The final comparison to be made in validating the simulation is comparison of relative error magnitudes for different controllers. For each joint, the errors increase in magnitude from the feedforward full to the feedforward diagonal to the PD controller. This is an accurate reflection of how the errors are expected to increase as you remove dynamic compensation.

Summary

In each case, the simulation errors are representative of the actual PUMA errors. It is important to recall that exact error matching cannot be expected from the reduced analog model. What is required is that the simulation give the control enginner comparative error magnitude and error trend information.

Joint one does a good job of giving error trend information, as well as error magnitudes. Joint two gives this information; however, when the feedforward full controller is used, the final position error information is somewhat disappointing. Joint three gives a very accurate representation of the actual PUMA.

Chapter Five

Conclusions and Recommendations

Summary of Results

The first real-time simulation of the positioning joints of an industrial manipulator has been developed. This simulation provides the capability of testing digital, analog, and hybrid control algorithms because the model runs in the analog section of the computer. Simulation of man-in-the-loop algorithms in real-time is also a capability provided by the simulation.

Previous simulations in digital computers were unable to test analog controllers in real-time because the digital simulation must be able to model the nonlinearities of analog systems and can not compute these complex functions fast enough. By programming the PUMA model in the analog portion of the SIMSTAR, the ability to test analog and analog/digital controllers is created. Standard digital controllers can also be tested. This capability provides the control engineer with the ability to test new modern control techniques prior to implementation on a robot arm. The capability to evaluate controllers in real time gives the engineer much more flexibility in designing a controller.

Previous to this research, man-in-the-loop research of teleoperated robots has been restricted to implementing a technique on a robot, testing, and correcting problems as they are found. With this real time simulation, the capability to test algorithms, and find optimum solutions, has been developed. The SIMSTAR has the capability of taking external analog signals into the analog portion for use in the simulation. This potential will allow a researcher to take analog signals from an input device, integrate it into the control scheme, and output information about joint positions in real time. One possible use of this would be to fix an exoskeleton to a person, input the joint information, and output the model's position on a graphic display terminal. Once again, this will allow the engineer to iterate his design using a simulation before implementing it on a remote control arm.

Limitations due to current hardware restrictions in the AFIT SIMSTAR also exist, although many of the ones previously mentioned will eventually be corrected. Software anomalies exist in the operating system that annoy the programmer/user, but these can all be worked around and are being eliminated through operating system updates.

Conclusions

The simulation does provide an accurate representation of the PUMA 560. Improvements can be made in terms of simplification of the model and in increasing the accuracy of joint two model. This simulation can be converted to model other robot arms by modification of the parameters. The basis Lagrange-Euler formulation is applicable to any robot. Modification of the dynamic parameters is all that is required to introduce a different robot model. Prior knowledge of the robot model is useful, but as this thesis showed, experimental evaluation is a necessity.

The SIMSTAR provides a unique environment in which to work. It allows analog programming in software, which relieves the researcher of the burden of patching the analog computer by hand. The interface between digital and analog portions, while it could be improved, is handled internally in the SIMSTAR. This feature is key to allowing hybrid controller research to be attempted.

This researcher found the SIMSTAR a very demanding environment to build a simulation in. The operating system is very user unfriendly, but that is more of an annoyance than a hindrance. The hindrance came in the form of anomalies in the software that would delay the implementation of a change by a factor of 10. In one case, it took four working days for this programmer to correct a minor problem with a digital counter. The code change required all of eight lines of code. See Appendix F for SIMSTAR hooks and handles.

Recommendations

Research using the simulation can branch in several directions. Further research into modern control techniques, as well as, hybrid controllers can be accomplished. To improve the simulations ability to test controllers, a communications processor (DCP) should be added to the SIMSTAR. This will allow testing different sampling rates as well as controllers.

By splitting the feedforward and feedback portions of a controller, research into sampling rates necessary for the feedforward dynamics compensation can be performed. By knowing the sampling rates necessary for both the feedforward and feedback loops to model the system accurately, the control engineer can allocate the available processing power more efficiently.

Different trajectory generators can also be investigated. By adjusting the scaling of the analog variables, saturation of actuators can be noted for any new trajectory generator.

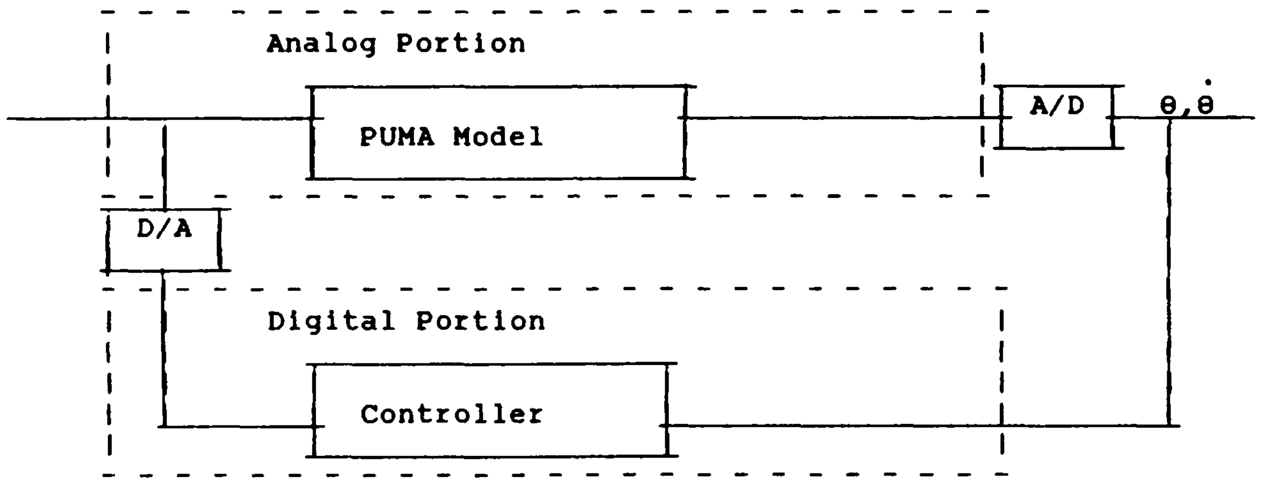
As discussed earlier, man-in-the-loop research of remote control arms is a possible future research area. To accomplish this, it would be necessary to integrate a graphics terminal, through the digital portion of the SIMSTAR, with the analog model. The graphic display would need the ability to display a three jointed arm that could be updated in real time. This would allow the researcher to get visual feedback on the arm's position.

Appendix A

Simulation Programs

Index of Programs

Figure A.1. Structure of Each Simulation Program



This program, S1.FFFG, contains the feedforward/full controller. The model of the PUMA is contained in the PARALLEL section of the program, being subdivided into calculation of the separate dynamic terms, and the actual Lagrange-Euler dynamics itself. The controller is in the DERIVATIVE section, with any loops or nonlinear terms being calculated in separate FORTRAN subroutines.

```
*PSP=1,0,ERR=ALL
*TITLE
S1.FFFG - MODEL OF 3 DOF PUMA 560
*INPUT
PROGRAM
```

```
      '    '
      '    '
   '      THIS PROGRAM SIMULATES THE FIRST THREE JOINTS OF '
   '  A PUMA 560 USING A MODEL DEVELOPED BY TARN, WITH STATIC  '
   '  AND VISCOUS FRICTION MODELLING ADDED ON.  THE ARM MODEL  '
   '  IS BASED ON LAGRANGE-EULER DYNAMICS, WITH INSIGNIFICANT TERMS '
   '  REMOVED.  THE MODEL       '
   '  EXISTS IN THE PARALLEL REGION OF THE SIMSTAR AND THE '
   '  CONTROLLER CAN BE PLACED EITHER IN THE DISCRETE OR    '
   ' PARALLEL REGION.  HINT: WATCH OUT FOR USING TOO MANY ADDS '
   ' OR MULTIPLIES IN THE PARALLEL REGION.                      '
   ' '
   '  WRITTEN BY :  CAPT PETER VAN WIRT       '
   '   '
   '  LAST CHANGED:  4 NOV 87 (PVW)     '
   ' '
   ' '
   ' '
  'INTERRUPT DECLARATIONS'
   ' INTDEF(0,1,1) '
   ' INTDEF(1,1,0) '
   '     '
   '     SCALING OF VARIABLES, SETTING CONSTANTS    '
   '     '
         '@SCALE  D12=2.62, D13=0.17, D23=1.03 '
         '@SCALE D122=2.14, D123=0.17, D133=0.17, D223=0.47'
         '@SCALE D233=0.47, G2=64, G3=10.78, Q1=2.8, Q2=3.93'
         '@SCALE D211=3, D311=3, D322=0.47'
         '@SCALE QC=3.93, QD1=2.25, QD2=1.6 , QDC=3.3 , QDD1=18.0'
         '@SCALE QDD2=19  ,QDD3=25 , QX=4.7, T1=73 , T2=90 , T3=36 '
         '@SCALE C2=1, S2=1, C3=1, S3=1, C23=1, S23=1'
         '@SCALE C2S23=1, C2C23=1, QD23=5.36, QDC3=10.1'
         '@SCALE T01=73, T02=90, T03=36'
         '@SCALE C2S2 = 1, S2S23 = 1, D11X = 2, D112 = 3, D113 = 3'
         '@SCALE STICK1=5.95, STICK2=6.82, STICK3=3.91'
         '@SCALE VISC1=10.13, VISC2=5.6, VISC3=10.9'
         '@SCALE ONE1=5.95, TWO1=6.82, THREE1=3.91 '
         '@SCALE ONE2=5.95, TWO2=6.82, THREE2=3.91 '
         '@SCALE TORQ1=85, TORQ2=100, TORQC = 41 '
         '@PARAMETER INIQD1,INIQD2,INIQDC,INITQ1,INITQ2'
         '@PARAMETER INITQC'
         '@MAXVAL INIQD1=2.25, INIQD2=1.6 , INIQDC=3.1, D22=6.37'
         '@MAXVAL INITQ1=2.8 , INITQ2=3.93, INITQC=3.2 , D11=6.12'
         '@MINVAL INIQD1=-2.25, INIQD2=-1.6, INIQDC=-3.1, D22=4.0'
         '@MINVAL INITQ1=-2.8, INITQ2=-3.2, INITQC=-3.93, D11=.5'
INITIAL
     '@BETA(BETA)'
     MAXT = PERIOD/BETA
     LOGPER = CINT * BETA
    '    '
    '   SET RUN CONTROL VARIABLES AND DEFINE VARIABLE TYPES  '
    '    '
    VARIABLE TIME = 0
     CONSTANT BETA =1, RUNTIM = 1.48, PERIOD =.01401
CONSTANT T1MAX=73, T2MAX=90, T3MAX=36, POINTR=.014
CONSTANT CINT=.014, KV1=75.12, KV2=156.4, KV3=28.66
CONSTANT KP1=563.4, KP2=1172, KP3= 215,A=2.3562,B=0
CONSTANT STATF1=5.95, STATF2=6.82, STATF3=3.91
CONSTANT ICT1=5.95,ICT2=43,ICT3=-3.7
CONSTANT ICSF1=5.95,ICSF2=6.82,ICSF3=-3.91
INTEGER NUM
'@PARAMETER BETA,  RUNTIM, POINTR '
'@MAXVAL BETA =100,  RUNTIM= 7, TIME=50, POINTR=.014'
'@MINVAL BETA =.1, RUNTIM=0, TIME=0,POINTR=.001'
ARRAY QEDD(3,720),QED(3,720),QE(3,720)
REAL D011,D022,D033,C22,S2233,C2233,S33,C33,...
     T01,T02,T03,VE1,VE2,VE3,PE1,PE2,PE3,D012,D013,D023, ...
     GG2,GG3,VF1,VF2,VF3,STF1,STF2,STF3
NSTEPS NSTP = 1
LOGICAL ST1, ST2, ST3
' '
'  LOAD THE DESIRED JOINT POSITIONS, VELOCITIES, AND  '
'  ACCELERATIONS.  THESE ARE IN FILES GENERATED BY A   '
'  SEPERATE PROGRAM CALLED S.TRAJEC  .                 '
'    '
CALL LOADING(QEDD,QED,QE,POINTR,A,B)
' '
 INIQD1 = 0
 INIQD2 = 0
 INIQDC = 0
 INITQ1 = B
 INITQ2 = -A
 INITQC = A
 VE1=0.0
 VE2=0.0
 VE3=0.0
 PE1=0.0
 PE2=0.0
 PE3=0.0
'        INITIAL TORQUE IS INPUT INTO THE ARM TO ACCOUNT FOR '
'   GRAVITY AND OTHER TERMS THAT EXIST PRIOR TO T=0.  '
'   VARIABLES ICT1,ICT2,ICT3 ARE USED SO THAT THIS INITIAL '
'   TORQUE CAN BE MODIFIED IN SIMRUN TO ACCOUNT FOR DIFFERENT '
'   INITIAL CONDITIONS.  '
'   '
 T01= ICT1
 T02= ICT2
 T03= ICT3
 QEDD(1,0)=0
 QEDD(2,0)=0
 QEDD(3,0)=0
 QED(1,0)=0
 QED(2,0)=0
 QED(3,0)=0
 QE(1,B)=B
 QE(2,0)= - A
 QE(3,0)= A
 NUM = 0
```

```
           D011 = 0
           D022 = 0
           D033 = 0
           D012 = 0
           D013 = 0
           D023 = 0
           VF1 = 0
           VF2 = 0
           VF3 = 0
       '  '
       '     INITIAL FRICITON IS INPUT INTO THE ARM TO ACCOUNT FOR '
       '  AMBIGUITY IN DIRECTION INITIALLY EXHIBITED.        '
       '  VARIABLES ICSF1,ICSF2,ICSF3 ARE USED SO THAT THIS INITIAL '
       '  FRICTION CAN BE MODIFIED IN SIMRUN TO ACCOUNT FOR DIFFERENT '
       '  INITIAL CONDITIONS.  '
       '  '
           STF1 = ICSF1
           STF2 = ICSF2
           STF3 = ICSF3
         END $'INITIAL'
    DYNAMIC
        'Interrupt Rate Error Declarations'
            LOGICAL ENDER1,RATER1,ERROR1
            ENDER1 = .FALSE.
            ERROR1 = RATER1
    DERIVATIVE
        '  '
       ' FEEDFORWARD CONTROL LAW    '
       ' I.E. THIS IS THE CONTROLLER  '
       '  '
       '  VARIABLES -                                             '
       '       D011,D022,D033 - DIAGONAL INERTIAL COMPONENTS      '
       '       D012,D013,D023 - OFF-DIAGONAL COMPONENTS           '
       '       T01,T02,T03 - JOINT TORQUES                        '
       '       QEDD(I,J) - JOINT ACCELERATIONS                         '
       '       QED(I,J) - DESIRED JOINT VELOCITIES                     '
       '       QDA1,QDA2,QDA3 - ACTUAL JOINT VELOCITIES           '
       '       QE(I,J) - DESIRED JOINT TRAJECTORIES                 '
       '       QA1,QA2,QA3 - ACTUAL JOINT TRAJECTORIES            '
       '       KV,KP - COEFFICIENTS USED TO POSITION THE          '
       '               CONTROLLERS POLES                         '
       '       VE1,VE2,VE3 - VELOCITY ERROR   '
       '       PE1,PE2,PE3 - POSITION ERROR   '
       '  '
        CALL INCRE(NUM,TIIME)
        C22 = COS(QA2)
        S22 = SIN(QA2)
        S2233 = SIN(QA2 + QA3)
        C2233 = COS(QA2 + QA3)
        S33 = SIN(QA3)
        C33 = COS(QA3)
        D011=2.4975 + 2.1007*C22**2 + 0.5323*S2233 - 0.033*C22*C2233 ...
              - 0.0405*C2233*S2233+ 0.9161*C22*S2233
        D022 = 5.419 + 0.9161*S33 - 0.0331*C33
```

```
     D033 =  1.1295
     D013 = -0.007*S2233 - 0.1596*C2233
     D012 = 2.4492 *S22 + D013
     D023 = 0.5468 + 0.4581*S33 - 0.0165*C33
     GG3 = 0.3761*C2233 - 10.4068*S2233
     GG2 = -52.106*C22 + 1.0972*S22 + GG3
     VF1 = 4.5 * QDA1
     VF2 = 3.5 * QDA2
     VF3 = 3.5 * QDA3
     '  '
     CALL STATIC(QDA1,QDA2,QDA3,T01,T02,T03,STF1,STF2,STF3)
     VE1 = QED(1,NUM) - QDA1
     VE2 = QED(2,NUM) - QDA2
     VE3 = QED(3,NUM) - QDA3
     PE1 = QE(1,NUM) - QA1
     PE2 = QE(2,NUM) - QA2
     PE3 = QE(3,NUM) - QA3
     T01 = D011*QEDD(1,NUM) + D012*QEDD(2,NUM) + D013*QEDD(3,NUM) ...
           + KV1* VE1 + KP1* PE1 + STF1 + VF1
     T02 = D022*QEDD(2,NUM) + D012*QEDD(1,NUM) + D023*QEDD(3,NUM) ...
           + KV2* VE2 + KP2* PE2 + STF2 + VF2 + GG2
     T03 = D033*QEDD(3,NUM) + D013*QEDD(1,NUM) + D023*QEDD(2,NUM) ...
           + KV3* VE3 + KP3* PE3 + STF3 + VF3 + GG3
  '    '
  '         THIS SECTION CONTAINS A/D CONVERSION EQUATIONS     '
  '    '
     QA1 = Q1
     QA2 = Q2
     QA3 = QC
     QDA1 = QD1
     QDA2 = QD2
     QDA3 = QDC
     TIIME=TIME
   '   '
   '        THIS SECTION CONTAINS EQUATIONS TO CONVERT ARRAY DATA '
   '    INTO VARIABLES SO THAT THEY CAN BE DISPLAYED USING THE '
   '    PREPAR STATEMENT IN SIMSTARS SIMRUN.      '
   '   '
     QED1 = QED(1,NUM)
     QED2 = QED(2,NUM)
     QED3 = QED(3,NUM)
     QE1 = QE(1,NUM)
     QE2 = QE(2,NUM)
     QE3 = QE(3,NUM)
    '      '
    '      '
    '      '
 '@PARALLEL'
     '        THIS REGION CONTAINS THE ANALOG MODEL.      '
     '   '
     '   '
    '     REDUCING COMPUTATIONAL LOADING BY PRODUCING VARIABLES  '
 '  THAT ARE USED MORE THAN ONCE IN THE PARALLEL REGION.  THIS '
 '  MINIMIZES THE NUMBER OF SUMMERS AND MULTIPLIERS NEEDED TO '
```

```
'  RUN THE MODEL.    '
'    '
  C2 = COS(Q2)
  S2 = SIN(Q2)
  C3 = COS(QC)
  S3 = SIN(QC)
  QX = Q2 + QC
  C23 = COS(QX)
  S23 = SIN(QX)
  C2S2 = C2*S2
  S2S23 = S2*S23
  C2S23 =  C2*S23
  C2C23 = C2*C23
  QD23 = QD2*QDC
  QDC3 = QDC*QDC
  ''
'  TORQUE VALUES COMPUTED FROM DERIVATIVE REGION '
  ''
  '  VARIABLES:                                        '
  '  '
  '     T1,T2,T3 - ANALOG VARIABLES OF JOINT TORQUES      '
  '     T01,T02,T03 - DIGITAL VARIABLES OF JOINT TORQUES  '
  '     '
  T1 = T01
  T2 = T02
  T3 = T03
  ''
'  CALCULATING MODEL DYNAMIC'S COEFFICIENTS '
  ''
  '     VARIABLES:                                   '
  '     '
  '     D"IJ" - THE ROW "I", COLUMN "J" COMPONENT OF THE  '
  '             INERTIA MATRIX                            '
  '     D"IJK" - THE ROW "I", COLUMN "J" , DEPTH "K" COMPONENT '
  '              OF THE THIRD ORDER CORIOLIS AND CETRIFUGAL TENSOR'
  '     G1,G2,G3 - GRAVITY COMPONENTS ... G1 = 0                '
  '     '
  D11 = 2.4975 + 2.1007*C2**2 + 0.5323*S23**2 ...
       + 0.9161*C2S23
  D22 = 5.419 + 0.9161*S3
  D12 = 2.4492*S2 + D13
  D13 = -0.007*S23 - 0.1596*C23
  D23 = 0.5468 + 0.4581*S3
  D11X = 0.5322*C3*S3 - 1.0643*S3*S2S23 + 0.4581*C2C23
  D112 = ( D11X - 1.5685*C2S2 - 0.4519*S2S23)
  D113 = ( D11X + 0.5322 * C2S2)
  D122 =   ( 1.9686*C2 + D123)
  D123 = ( 0.1596*S23 - 0.007*C23)
  D133 = D123
  D211 = - D112
  D223 = ( 0.4581*C3 + 0.0165*S3)
  D233 = D223
  D311 = - D113
  D322 = - D223
```

```
G2 = -52.106*C2 + 1.0972*S2 + G3
G3 = 0.3761*C23 - 10.4068*S23
'   '
'       THIS SECTION CALCULATES THE STATIC FRICTION OF EACH  '
' JOINT.  THE FRICTION IS A CONSTANT VALUE WHOSE SIGN IS '
' BASED ON THE SIGN OF THE JOINT VELOCITY.  IF JOINT  '
' VELOCITY IS BELOW A CERTAIN VALUE, THE SIGN OF THE   '
' FRICTION CONSTANT IS BASED ON THE DIRECTION OF APPLIED '
' TORQUE.   '
'   '
'  VARIABLES:   '
'      ST1,ST2,ST3 - LOGICAL VARIABLES USED TO DETERMINE '
'                 WHETHER TO USE VELOCITY OR TORQUE SIGN  '
'      ONE1,TWO1,THREE1 - PUTS OUT A (+/-)FRICTION BASED '
'                 ON DIRECTION OF TORQUE     '
'      ONE2,TWO2,THREE2 - PUTS OUT A (+/-)FRICTION BASED '
'                         ON DIRECTION OF VELOCITY   '
'      STICK1,STICK2,STICK3 - CHOSES PROPER FRICTION VALUE '
'                         BASED ON ST1,ST2,ST3   '
'     '
ST1 = ABS(QD1) .GT. 0.01
ST2 = ABS(QD2) .GT. 0.01
ST3 = ABS(QDC) .GT. 0.01
ONE1 = FCNSW(T1,-STATF1,0.0,STATF1)
ONE2 = FCNSW(QD1,-STATF1,0.0,STATF1)
TWO1 = FCNSW(T2,-STATF2,0.0,STATF2)
TWO2 = FCNSW(QD2,-STATF2,0.0,STATF2)
THREE1 = FCNSW(T3,-STATF3,0.0,STATF3)
THREE2 = FCNSW(QDC,-STATF3,0.0,STATF3)
STICK1 = RSW(ST1,ONE2,ONE1)
STICK2 = RSW(ST2,TWO2,TWO1)
STICK3 = RSW(ST3,THREE2,THREE1)
'   '
'       THIS SECTION CALCULATES THE VISCOUS FRICTION OF EACH '
' JOINT.  IT IS A CONSTANT VALUE TIMES THE VELOCITY OF THE   '
' JOINT.  '
'  '
VISC1 = 4.5 * QD1
VISC2 = 3.5 * QD2
VISC3 = 3.5 * QDC
'   '
'    ADDITIVE TERMS IN THE MODEL.  THIS AVIODS A CODING PROBLEM'
'  ENCOUNTERED IN THE '@IMPL' REGION.  P-TRAN CONSIDERS ALL OF '
'  THE EQUATIONS IN THAT REGION AS ONE.  THERE ARE TOO MANY   '
'  VARIABLES ON THE RIGHT HAND SIDE IN THAT SECTION.   '
'  '
TORQ1 = - T1 + STICK1 + VISC1
TORQ2 = - T2 + STICK2 + VISC2 + G2 + D211*QD1*QD1
TORQC = - T3 + STICK3 + VISC3 + G3 + D311*QD1*QD1 ...
        + D322*QD2*QD2
''
'  MODEL DYNAMIC EQUATIONS '
''
'  VARIABLES:                                             '
```

```
    '  '
    '  QDD1,QDD2,QDD3 - JOINT ACCELERATIONS                       '
    '  QD1,QD2,QDC -  JOINT VELOCITIES                            '
    '  Q1,Q2,QC -  JOINT POSITIONS                                '
    '  '
    '      THIS NEXT SECTION CONTAINS AN ARITHMATIC LOOP CAUSED '
    '  BY THE COUPLED NATURE OF THE MODEL.  THE PROCEDURAL AND  '
    '   IMPL ARE NECESSARY TO INSTRUCT P-TRAN IN HANDLING THE   '
    '  SITUATION.   '
    '  '
    '  '
   PROCEDURAL (QDD1,QDD2,QDD3  = D12,D13,D122,QD2 ...
         ,D123,QD23,D133,QDC3,D11,D23,D223 ...
         ,D233,QDC3,D22,TORQ1,TORQ2,TORQC)
   '@IMPL (QDD3,QDD2)'
   QDD1 = -( TORQ1 + D12*QDD2 + D13*QDD3 + D122*QD2*QD2 ...
          + 2*D123*QD23 + D133*QDC3 + 2*D112*QD1*QD2 ...
          + 2*D113*QD1*QDC)/D11
   QDD2 = -( TORQ2 + D23*QDD3 + 2*D223*QD23 + D233*QDC3 ...
               + D12 * QDD1 )/D22
   QDD3 = -0.88535 * ( TORQC  + D13 * QDD1 ...
          + D23 * QDD2)
   '@END IMPL'
   END
   QD1  = INTEG(QDD1,INIQD1)
   QD2  = INTEG(QDD2,INIQD2)
   QDC  = INTEG(QDD3,INIQDC)
   Q1   = INTEG(QD1,INITQ1)
   Q2   = INTEG(QD2,INITQ2)
   QC   = INTEG(QDC,INITQC)
       TERMT(TIME .GT. RUNTIM)
   '    '
   'DEFINE INTERRUPT CONTROL'
   '  '
      LOGICAL GPI0,GPI1
      GPI0 = CLOCK(PERIOD)
      GPI1 = CLOCK(LOGPER)
      '@INTRRT 1 =GPI0'
      '@INTRRT 2 =GPI1'
       RATER1 = RATERR(GPI0,ENDER1)
      '@RECORD(REC01,,,,,,,,,,,)'
      '@ENDPARALLEL'
 END $ 'OF DERIVATIVE'
 END $ 'OF DYNAMIC'
 TERMINAL $ END $ 'OF TERMINAL'
 END $ 'OF PROGRAM'
 *TRANSLATE
    '  '
    '  SET UP A/D AND D/A CONVERTERS  '
    '  '
     DCA(1) = T01,T02,T03
     PADC(1) = Q1,Q2,QC,QD1,QD2,QDC,TIME
 *OUTPUT
 *END
```

```
      SUBROUTINE PREP1
+
   INCLUDE E1.FFFG
      Q1 = QRPADC(0)*S:Q1
      Q2 = QRPADC(1)*S:Q2
      QC = QRPADC(2)*S:QC
      QD1 = QRPADC(3)*S:QD1
      QD2 = QRPADC(4)*S:QD2
      QDC = QRPADC(5)*S:QDC
     TIME = QRPADC(6)*S:TIME
      RETURN
      END
C
      SUBROUTINE POST1
+
   INCLUDE E1.FFFG
      COMMON /QQDCP/DCASF(0:2)
      LOGICAL DELAY
      CALL QWDCAR(0,T01*DCASF(0))
      CALL QWDCAR(1,T02*DCASF(1))
      CALL QWDCAR(2,T03*DCASF(2))
      IF (L:RATER1) CALL ZZRTER(1)
      L:ENDER1 = .TRUE.
      DELAY = L:ENDER1
      L:ENDER1 = .FALSE.
      RETURN
      END
C
      SUBROUTINE PREPDCA
+
      COMMON /QQDCP/DCASF(0:2)
      DCASF(0) = 1.0/QDCASR(0)/T1MAX
      DCASF(1) = 1.0/QDCASR(1)/T2MAX
      DCASF(2) = 1.0/QDCASR(2)/T3MAX
      RETURN
      END
C
      SUBROUTINE LOADING(QEDD,QED,QE,POINTER,A,B)
      DIMENSION QEDD(3,720),QED(3,720),QE(3,720)
      REAL TF,POINTER,INPOS1,INPOS2,INPOS3
      INTEGER NUMPTS,STAT1,STAT2,STAT3
      TF=1.5
      INPOS1 = B
      INPOS2 = -A
      INPOS3 = A
      NUMPTS=IFIX(TF/0.007) + 1
      OPEN(UNIT=13,
     1 OPENMODE='R',BLOCKED=.TRUE.)
      OPEN(UNIT=11,
     1 OPENMODE='R',BLOCKED=.TRUE.)
      OPEN(UNIT=12,
     1 OPENMODE='R',BLOCKED=.TRUE.)
      DO 300 J=1,NUMPTS
      READ(13,400) (QE(I,J),I=1,3)
```

```
          READ(11,400) (QED(I,J),I=1,3)
          READ(12,400) (QEDD(I,J),I=1,3)
          QE(1,J) = QE(1,J) + INPOS1
          QE(2,J) = QE(2,J) + INPOS2
          QE(3,J) = QE(3,J) + INPOS3
          QED(3,J) = QED(3,J)
          QEDD(3,J) = QEDD(3,J)
300       CONTINUE
400       FORMAT(3(E12.6))
          CLOSE(UNIT=13,STATUS='KEEP')
          CLOSE(UNIT=11,STATUS='KEEP')
          CLOSE(UNIT=12,STATUS='KEEP')
          RETURN
          END
         SUBROUTINE STATIC(QDA1,QDA2,QDA3,T01,T02,T03,STF1,STF2,STF3)
          REAL QDA1,QDA2,QDA3,T01,T02,T03,STF1,STF2,STF3
             IF (ABS(QDA1) .GT. 0.01) THEN
                   STF1 = SIGN(5.95,QDA1)
             ELSE
                   STF1 = SIGN(5.95,T01)
             ENDIF
             IF (ABS(QDA2) .GT. 0.01) THEN
                   STF2 = SIGN(6.82,QDA2)
             ELSE
                   STF2 = SIGN(6.82,T02)
             ENDIF
             IF (ABS(QDA3) .GT. 0.01) THEN
                   STF3 = SIGN(3.91,QDA3)
             ELSE
                   STF3 = SIGN(3.91,T03)
             ENDIF
          RETURN
          END
          SUBROUTINE INCRE(DNUM,DTIIME)
          INTEGER DNUM,STEP
          REAL DTIIME
          STEP = 2
          IF (DTIIME .LE. 0.021) THEN
               DNUM = 1
          ELSE
               DNUM = DNUM + STEP
          ENDIF
          RETURN
          END
```

This program, S1.FFDG, contains the feedforward/diagonal controller. The model of the PUMA is contained in the PARALLEL section of the program, being subdivided into calculation of the separate dynamic terms, and the actual Lagrange-Euler dynamics itself. The controller is in the DERIVATIVE section, with any loops or nonlinear terms being calculated in separate FORTRAN subroutines.

```
*PSP=1,0,ERR=ALL
*TITLE
S1.FFDG - MODEL OF 3 DOF PUMA 560
*INPUT
PROGRAM
      '    '
      '    '
      '    '
   '     THIS PROGRAM SIMULATES THE FIRST THREE JOINTS OF '
   '  A PUMA 560 USING A MODEL DEVELOPED BY TARN.  THE ARM MODEL '
   '  IS BASED ON LAGRANGE-EULER DYNAMICS, WITH INSIGNIFICANT TERMS '
   '  REMOVED.  THE MODEL      '
   '  EXISTS IN THE PARALLEL REGION OF THE SIMSTAR AND THE '
   '  CONTROLLER CAN BE PLACED EITHER IN THE DISCRETE OR    '
   ' PARALLEL REGION.  HINT: WATCH OUT FOR USING TOO MANY ADDS '
   ' OR MULTIPLIES IN THE PARALLEL REGION.                     '
   ' '
   '  WRITTEN BY :  CAPT PETER VAN WIRT      '
   '  '
   '  LAST CHANGED: 4 NOV 87 (PVW)     '
   ' '
   ' '
   ' '
  'INTERRUPT DECLARATIONS'
   ' INTDEF(0,1,1) '
   ' INTDEF(1,1,0) '
   '    '
   '    SCALING OF VARIABLES, SETTING CONSTANTS   '
   '    '
        '@SCALE  D12=2.62, D13=0.17, D23=1.03 '
        '@SCALE D122=2.14, D123=0.17, D133=0.17, D223=0.47'
        '@SCALE D233=0.47, G2=64, G3=10.78, Q1=2.8, Q2=3.93'
        '@SCALE D211=3, D311= 3, D322= 0.47'
        '@SCALE Q3=3.93, QD1=2.25, QD2=1.6 , QD3=3.3 , QDD1=18.0'
        '@SCALE QDD2=19  ,QDD3=25 , QX=4.7, T1=73 , T2=90 , T3=36 '
        '@SCALE C2=1, S2=1, C3=1, S3=1, C23=1, S23=1'
        '@SCALE C2S23=1, C2C23=1, QD23=5.36, QD33=10.1'
        '@SCALE T01=73, T02=90, T03=36'
        '@SCALE C2S2=1, S2S23=1, D11X=2, D112=3, D113=3'
        '@SCALE STICK1=5.95, STICK2=6.82, STICK3=3.91'
        '@SCALE VISC1=10.13, VISC2=5.6, VISC3=10.9'
```

```
          '@SCALE ONE1=5.95, TWO1=6.82, THREE1=3.91 '
          '@SCALE ONE2=5.95, TWO2=6.82, THREE2=3.91 '
          '@SCALE TORQ1=85, TORQ2=100, TORQ3 = 41 '
          '@PARAMETER INIQD1,INIQD2,INIQD3,INITQ1,INITQ2'
          '@PARAMETER INITQ3'
          '@MAXVAL INIQD1=2.25, INIQD2=1.6 , INIQD3=3.1, D22=6.37'
          '@MAXVAL INITQ1=2.8 , INITQ2=3.93, INITQ3=3.2 , D11=6.12'
          '@MINVAL INIQD1=-2.25, INIQD2=-1.6, INIQD3=-3.1, D22=4.0'
          '@MINVAL INITQ1=-2.8, INITQ2=-3.2, INITQ3=-3.93, D11=.5'
INITIAL
     '@BETA(BETA)'
     MAXT = PERIOD/BETA
     LOGPER = CINT * BETA
    '   '
    '   SET RUN CONTROL VARIABLES AND DEFINE VARIABLE TYPES  '
    '   '
    VARIABLE TIME = 0
     CONSTANT BETA =1, RUNTIM = 1.48, PERIOD =.01401
     CONSTANT T1MAX=73, T2MAX=90, T3MAX=36, POINTR=.014
     CONSTANT CINT=.014, KV1=75.12, KV2=156.4, KV3=28.66
     CONSTANT KP1=563.4, KP2=1172, KP3=215,A=2.3562,B=0
     CONSTANT STATF1=5.95, STATF2=6.82, STATF3=3.91
     CONSTANT ICT1=5.95,ICT2=43,ICT3=-3.7
     CONSTANT ICSF1=5.95,ICSF2=6.82,ICSF3=-3.91
     INTEGER NUM
     '@PARAMETER BETA,  RUNTIM, POINTR '
     '@MAXVAL BETA =100,  RUNTIM= 7, TIME=50, POINTR=.01401'
     '@MINVAL BETA =.1, RUNTIM=0, TIME=0,POINTR=.001'
     ARRAY QEDD(3,720),QED(3,720),QE(3,720)
     REAL D011,D022,D033,C22,S2233,C2233,S33,C33,...
          T01,T02,T03,VE1,VE2,VE3,PE1,PE2,PE3,D012,D013,D023, ...
          GG2,GG3,VF1,VF2,VF3,STF1,STF2,STF3
     NSTEPS NSTP = 1
     LOGICAL ST1, ST2, ST3
     ' '
     '  LOAD THE DESIRED JOINT POSITIONS, VELOCITIES, AND  '
     '  ACCELERATIONS.  THESE ARE IN FILES GENERATED BY A   '
     '  SEPERATE PROGRAM CALLED S.TRAJEC  .                 '
     '   '
     CALL LOADING(QEDD,QED,QE,POINTR,A,B)
    ' '
      INIQD1 = 0
      INIQD2 = 0
      INIQD3 = 0
      INITQ1 = B
      INITQ2 = -A
      INITQ3 = A
      VE1=0.0
      VE2=0.0
      VE3=0.0
      PE1=0.0
      PE2=0.0
      PE3=0.0
 '  '
```

```
      '     INITIAL TORQUE IS INPUT INTO THE ARM TO ACCOUNT FOR '
      '  GRAVITY AND OTHER TERMS THAT EXIST PRIOR TO T=0.  '
      '  VARIABLES ICT1,ITC2,ITC3 ARE USED SO THAT THIS INITIAL  '
      '  TORQUE CAN BE MODIFIED IN SIMRUN TO ACCOUNT FOR DIFFERENT '
      '  INITIAL CONDITIONS.   '
      ' '
         T01= ICT1
         T02= ICT2
         T03= ICT3
         QEDD(1,0)=0
         QEDD(2,0)=0
         QEDD(3,0)=0
         QED(1,0)=0
         QED(2,0)=0
         QED(3,0)=0
         QE(1,B)=B
         QE(2,0)= - A
         QE(3,0)= A
         NUM = 0
         D011 = 0
         D022 = 0
         D033 = 0
         D012 = 0
         D013 = 0
         D023 = 0
         VF1 = 0
         VF2 = 0
         VF3 = 0
       ' '
       '  INITIAL STATIC VALUES ARE BASED ON INITIAL ROBOT  '
       ' POSITION.  '
       ' '
         STF1 = ICSF1
         STF2 = ICSF2
         STF3 = ICSF3
        END $'INITIAL'
DYNAMIC
   'Interrupt Rate Error Declarations'
          LOGICAL ENDER1,RATER1,ERROR1
          ENDER1 = .FALSE.
          ERROR1 = RATER1
DERIVATIVE
   ' '
       '  FEEDFORWARD CONTROL WITH DIAGONAL INERTIAL,FRICTION, AND '
       '  GRAVITY COMPENSATION.  '
       ' I.E. THIS IS THE CONTROLLER  '
       '  '
       '  VARIABLES -                                             '
       '       D011,D022,D033 - DIAGONAL INERTIAL COMPONENTS      '
       '       D012,D013,D023 - OFF-DIAGONAL COMPONENTS           '
       '       T01,T02,T03 - JOINT TORQUES                        '
       '       QEDD(I,J) - JOINT ACCELERATIONS                        '
       '       QED(I,J) - DESIRED JOINT VELOCITIES                    '
       '       QDA1,QDA2,QDA3 - ACTUAL JOINT VELOCITIES           '
       '       QE(I,J) - DESIRED JOINT TRAJECTORIES                 '
       '       QA1,QA2,QA3 - ACTUAL JOINT TRAJECTORIES            '
       '       KV,KP - COEFFICIENTS USED TO POSITION THE          '
       '               CONTROLLERS POLES                         '
       '       VE1,VE2,VE3 - VELOCITY ERROR   '
       '       PE1,PE2,PE3 - POSITION ERROR   '
       '  '
        CALL INCRE(NUM,TIIME)
        C22 = COS(QA2)
        S22 = SIN(QA2)
        S2233 = SIN(QA2 + QA3)
        C2233 = COS(QA2 + QA3)
        S33 = SIN(QA3)
        C33 = COS(QA3)
        D011=2.4975 + 2.1007*C22**2 + 0.5323*S2233 - 0.033*C22*C2233 ...
              - 0.0405*C2233*S2233+ 0.9161*C22*S2233
        D022 = 5.419 + 0.9161*S33 - 0.0331*C33
        D033 =  1.1295
        D013 = -0.007*S2233 - 0.1596*C2233
        D012 = 2.4492 *S22 + D013
        D023 = 0.5468 + 0.4581*S33 - 0.0165*C33
        GG3 = 0.3761*C2233 - 10.4068*S2233
        GG2 = -52.106*C22 + 1.0972*S22 + GG3
        VF1 = 4.5 * QDA1
        VF2 = 3.5 * QDA2
        VF3 = 3.5 * QDA3
        CALL STATIC(QDA1,QDA2,QDA3,T01,T02,T03,STF1,STF2,STF3)
        VE1 = QED(1,NUM) - QDA1
        VE2 = QED(2,NUM) - QDA2
        VE3 = QED(3,NUM) - QDA3
        PE1 = QE(1,NUM) - QA1
        PE2 = QE(2,NUM) - QA2
        PE3 = QE(3,NUM) - QA3
        T01 = D011*QEDD(1,NUM) ...
              + KV1 * VE1 + KP1 * PE1 + STF1 + VF1
        T02 = D022*QEDD(2,NUM) ...
              + KV2 * VE2 + KP2 * PE2 + STF2 + VF2 + GG2
        T03 = D033*QEDD(3,NUM) ...
              + KV3 * VE3 + KP3 * PE3 + STF3 + VF3 + GG3
       '  '
       '      THIS SECTION CONTROLS THE A/D CONVERSIONS.  '
       '  '
        QA1 = Q1
        QA2 = Q2
        QA3 = Q3
        QDA1 = QD1
        QDA2 = QD2
        QDA3 = QD3
        TIIME=TIME
      '     '
      '     '
      '     '
    '@PARALLEL'
        '        THIS REGION CONTAINS THE ANALOG MODEL.    '
    '    '
    '    '
    '    REDUCING COMPUTATIONAL LOADING BY PRODUCING VARIABLES  '
'  THAT ARE USED MORE THAN ONCE IN THE PARALLEL REGION.  THIS '
'  MINIMIZES THE NUMBER OF SUMMERS AND MULTIPLIERS NEEDED TO '
'  RUN THE MODEL.    '
'    '
 C2 = COS(Q2)
 S2 = SIN(Q2)
 C3 = COS(Q3)
 S3 = SIN(Q3)
 QX = Q2 + Q3
 C23 = COS(QX)
 S23 = SIN(QX)
 C2S23 =  C2*S23
 C2C23 = C2*C23
 C2S2 = C2*S2
 S2S23 = S2*S23
 QD23 = QD2*QD3
 QD33 = QD3*QD3
 ''
'  TORQUE VALUES COMPUTED FROM DERIVATIVE REGION '
 '  VARIABLES:                                            '
 '  '
 '     T1,T2,T3 - ANALOG VARIABLES OF JOINT TORQUES        '
 '     T01,T02,T03 - DIGITAL VARIABLES OF JOINT TORQUES    '
 '    '
 T1 = T01
 T2 = T02
 T3 = T03
 ''
'  CALCULATING MODEL DYNAMIC'S COEFFICIENTS '
 ''
 '     VARIABLES:                                      '
 '      '
 '     D"IJ" - THE ROW "I", COLUMN "J" COMPONENT OF THE  '
 '             INERTIA MATRIX                             '
 '     D"IJK" - THE ROW "I", COLUMN "J" , DEPTH "K" COMPONENT '
 '              OF THE THIRD ORDER CORIOLIS AND CETRIFUGAL TENSOR'
 '     G1,G2,G3 - GRAVITY COMPONENTS ... G1 = 0                  '
 '      '
 D11 = 2.4975 + 2.1007*C2**2 + 0.5323*S23**2 ...
        + 0.9161*C2S23
 D22 = 5.419 + 0.9161*S3
 D12 = 2.4492*S2 + D13
 D13 = -0.007*S23 - 0.1596*C23
 D23 = 0.5468 + 0.4581*S3
 D11X = 0.5322*C3*S3 - 1.0643*S3*S2S23 + 0.4581*C2C23
 D112 = ( D11X - 1.5685*C2S2 - 0.4519*S2S23)
 D113 = ( D11X + 0.5322*C2S2)
 D122 = 1.9686*C2 + D123
 D123 = ( 0.1596*S23 - 0.007*C23)
 D133 = D123
 D211 = - D112
```

```
D223 = ( 0.4581*C3 + 0.0165*S3)
D233 = D223
D311 = - D113
D322 = - D223
G2 = -52.106*C2 + 1.0972*S2 + G3
G3 = 0.3761*C23 - 10.4068*S23
'    '
'       THIS SECTION CALCULATES THE STATIC FRICTION OF EACH  '
' JOINT.  THE FRICTION IS A CONSTANT VALUE WHOSE SIGN IS '
' BASED ON THE SIGN OF THE JOINT VELOCITY.  IF JOINT  '
' VELOCITY IS BELOW A CERTAIN VALUE, THE SIGN OF THE   '
' FRICTION CONSTANT IS BASED ON THE DIRECTION OF APPLIED '
' TORQUE.  '
'    '
'  VARIABLES:   '
'      ST1,ST2,ST3 - LOGICAL VARIABLES USED TO DETERMINE '
'                 WHETHER TO USE VELOCITY OR TORQUE SIGN  '
'      ONE1,TWO1,THREE1 - PUTS OUT A (+/-)FRICTION BASED '
'                 ON DIRECTION OF TORQUE   '
'      ONE2,TWO2,THREE2 - PUTS OUT A (+/-)FRICTION BASED '
'                         ON DIRECTION OF VELOCITY   '
'      STICK1,STICK2,STICK3 - CHOSES PROPER FRICTION VALUE '
'                         BASED ON ST1,ST2,ST3   '
'    '
ST1 = ABS(QD1) .GT. 0.01
ST2 = ABS(QD2) .GT. 0.01
ST3 = ABS(QD3) .GT. 0.01
ONE1 = FCNSW(T1,-STATF1,0.0,STATF1)
ONE2 = FCNSW(QD1,-STATF1,0.0,STATF1)
TWO1 = FCNSW(T2,-STATF2,0.0,STATF2)
TWO2 = FCNSW(QD2,-STATF2,0.0,STATF2)
THREE1 = FCNSW(T3,-STATF3,0.0,STATF3)
THREE2 = FCNSW(QD3,-STATF3,0.0,STATF3)
STICK1 = RSW(ST1,ONE2,ONE1)
STICK2 = RSW(ST2,TWO2,TWO1)
STICK3 = RSW(ST3,THREE2,THREE1)
'    '
'       THIS SECTION CALCULATES THE VISCOUS FRICTION OF EACH '
' JOINT.  IT IS A CONSTANT VALUE TIMES THE VELOCITY OF THE   '
' JOINT.  '
'   '
VISC1 = 4.5 * QD1
VISC2 = 3.5 * QD2
VISC3 = 3.5 * QD3
'    '
'    ADDITIVE TERMS IN THE MODEL.  THIS AVIODS A CODING PROBLEM'
'  ENCOUNTERED IN THE '@IMPL' REGION.  P-TRAN CONSIDERS ALL OF '
'  THE EQUATIONS IN THAT REGION AS ONE.  THERE ARE TOO MANY   '
'  VARIABLES ON THE RIGHT HAND SIDE IN THAT SECTION.  '
'    '
TORQ1 = - T1 + STICK1 + VISC1
TORQ2 = - T2 + STICK2 + VISC2 + G2 + D211*QD1 *QD1
TORQ3 = - T3 + STICK3 + VISC3 + G3 + D311*QD1*QD1 ...
        + D322*QD2*QD2
```

```
  ''
 '  MODEL DYNAMIC EQUATIONS '
  ''
  '  VARIABLES:                                                  '
  '  '
  '  QDD1,QDD2,QDD3 - JOINT ACCELERATIONS                       '
  '  QD1,QD2,QD3 -  JOINT VELOCITIES                             '
  '  Q1,Q2,Q3 -  JOINT POSITIONS                                 '
  '  '
  '      THIS NEXT SECTION CONTAINS AN ARITHMATIC LOOP CAUSED '
  '  BY THE COUPLED NATURE OF THE MODEL.  THE PROCEDURAL AND  '
  '   IMPL ARE NECESSARY TO INSTRUCT P-TRAN IN HANDLING THE    '
  '  SITUATION.  '
  '  '
  '  '
 PROCEDURAL (QDD1,QDD2,QDD3  = D12,D13,D122,QD2 ...
      ,D123,QD23,D133,QD33,D11,D23,D223 ...
      ,D233,QD33,D22,TORQ1,TORQ2,TORQ3)
 '@IMPL (QDD3,QDD2)'
 QDD1 = -( TORQ1 + D12*QDD2 + D13*QDD3 + D122*QD2*QD2 ...
     + 2*D123*QD23 + D133*QD33 +2*D112*QD1*QD2 ...
     + 2*D113*QD1*QD3)/D11
 QDD2 = -( TORQ2 + D23*QDD3 + 2*D223*QD23 + D233*QD33 ...
             + D12 * QDD1)/D22
 QDD3 = -0.88535 * ( TORQ3  + D13 * QDD1 ...
        + D23 * QDD2)
 '@END IMPL'
 END
 QD1   = INTEG(QDD1,INIQD1)
 QD2   = INTEG(QDD2,INIQD2)
 QD3   = INTEG(QDD3,INIQD3)
 Q1    = INTEG(QD1,INITQ1)
 Q2    = INTEG(QD2,INITQ2)
 Q3    = INTEG(QD3,INITQ3)
     TERMT(TIME .GT. RUNTIM)
  '    '
 'DEFINE INTERRUPT CONTROL'
  '  '
    LOGICAL GPI0,GPI1
    GPI0 = CLOCK(PERIOD)
    GPI1 = CLOCK(LOGPER)
    '@INTRRT 1 =GPI0'
    '@INTRRT 2 =GPI1'
     RATER1 = RATERR(GPI0,ENDER1)
    '@RECORD(REC01,,,,,,,,,,,)'
    '@ENDPARALLEL'
END $ 'OF DERIVATIVE'
END $ 'OF DYNAMIC'
TERMINAL $ END $ 'OF TERMINAL'
END $ 'OF PROGRAM'
*TRANSLATE
  '  '
  '  SET UP A/D AND D/A CONVERTERS  '
  '  '
```

```
      DCA(1) = T01,T02,T03
      PADC(1) = Q1,Q2,Q3,QD1,QD2,QD3,TIME
*OUTPUT
*END
           SUBROUTINE PREP1
+
        INCLUDE E1.FFDG
           Q1 = QRPADC(0)*S:Q1
           Q2 = QRPADC(1)*S:Q2
           Q3 = QRPADC(2)*S:Q3
           QD1 = QRPADC(3)*S:QD1
           QD2 = QRPADC(4)*S:QD2
           QD3 = QRPADC(5)*S:QD3
          TIME = QRPADC(6)*S:TIME
           RETURN
           END
C
           SUBROUTINE POST1
+
        INCLUDE E1.FFDG
           COMMON /QQDCP/DCASF(0:2)
           LOGICAL DELAY
           CALL QWDCAR(0,T01*DCASF(0))
           CALL QWDCAR(1,T02*DCASF(1))
           CALL QWDCAR(2,T03*DCASF(2))
           IF (L:RATER1) CALL ZZRTER(1)
           L:ENDER1 = .TRUE.
           DELAY = L:ENDER1
           L:ENDER1 = .FALSE.
           RETURN
           END
C
           SUBROUTINE PREPDCA
+
           COMMON /QQDCP/DCASF(0:2)
           DCASF(0) = 1.0/QDCASR(0)/T1MAX
           DCASF(1) = 1.0/QDCASR(1)/T2MAX
           DCASF(2) = 1.0/QDCASR(2)/T3MAX
           RETURN
           END
C
           SUBROUTINE LOADING(QEDD,QED,QE,POINTER,A,B)
           DIMENSION QEDD(3,720),QED(3,720),QE(3,720)
           REAL TF,POINTER,INPOS1,INPOS2,INPOS3
           INTEGER NUMPTS,STAT1,STAT2,STAT3
           TF=1.5
           INPOS1 = B
           INPOS2 = -A
           INPOS3 = A
           NUMPTS=IFIX(TF/0.007) + 1
           OPEN(UNIT=13,
       1   OPENMODE='R',BLOCKED=.TRUE.)
           OPEN(UNIT=11,
       1   OPENMODE='R',BLOCKED=.TRUE.)
```

```
          OPEN(UNIT=12,
     1    OPENMODE='R',BLOCKED=.TRUE.)
          DO 300 J=1,NUMPTS
          READ(13,400) (QE(I,J),I=1,3)
          READ(11,400) (QED(I,J),I=1,3)
          READ(12,400) (QEDD(I,J),I=1,3)
          QE(1,J) = QE(1,J) + INPOS1
          QE(2,J) = QE(2,J) + INPOS2
          QE(3,J) = QE(3,J) + INPOS3
          QED(3,J) = QED(3,J)
          QEDD(3,J) = QEDD(3,J)
 300      CONTINUE
 400      FORMAT(3(E12.6))
          CLOSE(UNIT=13,STATUS='KEEP')
          CLOSE(UNIT=11,STATUS='KEEP')
          CLOSE(UNIT=12,STATUS='KEEP')
          RETURN
          END
        SUBROUTINE STATIC(QDA1,QDA2,QDA3,T01,T02,T03,STF1,STF2,STF3)
          REAL QDA1,QDA2,QDA3,T01,T02,T03,STF1,STF2,STF3
             IF (ABS(QDA1) .GT. 0.01) THEN
                    STF1 = SIGN(5.95,QDA1)
             ELSE
                    STF1 = SIGN(5.95,T01)
             ENDIF
             IF (ABS(QDA2) .GT. 0.01) THEN
                    STF2 = SIGN(6.82,QDA2)
             ELSE
                    STF2 = SIGN(6.82,T02)
             ENDIF
             IF (ABS(QDA3) .GT. 0.01) THEN
                    STF3 = SIGN(3.91,QDA3)
             ELSE
                    STF3 = SIGN(3.91,T03)
             ENDIF
          RETURN
          END
          SUBROUTINE INCRE(DNUM,DTIIME)
          INTEGER DNUM,STEP
          REAL DTIIME
          STEP = 2
          IF (DTIIME .LE. 0.021) THEN
            DNUM = 1
           ELSE
           DNUM = DNUM + STEP
          ENDIF
          RETURN
          END
```

This program, S1.PDG, contains the PD controller. The model of the PUMA is contained in the PARALLEL section of the program, being subdivided into calculation of the separate dynamic terms, and the actual Lagrange-Euler dynamics itself. The controller is in the DERIVATIVE section, with any loops or nonlinear terms being calculated in separate FORTRAN subroutines.

```
*PSP=1,0,ERR=ALL
*TITLE
S1.PDG - MODEL OF 3 DOF PUMA 560
*INPUT
PROGRAM
        '    '
        '    '
        '    '
     '      THIS PROGRAM SIMULATES THE FIRST THREE JOINTS OF '
     '  A PUMA 560 USING A MODEL DEVELOPED BY TARN.  THE ARM MODEL '
     '  IS BASED ON LAGRANGE-EULER DYNAMICS, WITH INSIGNIFICANT TERMS '
     '  REMOVED.  THE MODEL     '
     '  EXISTS IN THE PARALLEL REGION OF THE SIMSTAR AND THE '
     '  CONTROLLER CAN BE PLACED EITHER IN THE DISCRETE OR    '
     ' PARALLEL REGION.  HINT: WATCH OUT FOR USING TOO MANY ADDS '
     ' OR MULTIPLIES IN THE PARALLEL REGION.                     '
     ' '
     '  WRITTEN BY :  CAPT PETER VAN WIRT      '
     '  '
     '  LAST CHANGED:   4 NOV 87 (PVW)      '
     ' '
     ' '
     ' '
    'INTERRUPT DECLARATIONS'
     ' INTDEF(0,1,1) '
     ' INTDEF(1,1,0) '
     '     '
     '     SCALING OF VARIABLES, SETTING CONSTANTS   '
     '     '
           '@SCALE  D12=2.62, D13=0.17, D23=1.03 '
           '@SCALE D122=2.14, D123=0.17, D133=0.17, D223=0.47'
           '@SCALE D233=0.47, G2=64, G3=10.78, Q1=2.8, Q2=3.93'
           '@SCALE D211= 3, D311= 3, D322= 0.47'
           '@SCALE Q3=3.93, QD1=2.25, QD2=1.6 , QD3=3.3 , QDD1=18.0'
           '@SCALE QDD2=19  ,QDD3=25 , QX=4.7, T1=73 , T2=90 , T3=36 '
           '@SCALE C2=1, S2=1, C3=1, S3=1, C23=1, S23=1'
           '@SCALE C2S23=1, C2C23=1, QD23=5.36, QD33=10.1'
           '@SCALE T01=73, T02=90, T03=36'
           '@SCALE C2S2=1, S2S23=1, D11X=2, D112=3, D113=3'
           '@SCALE STICK1=5.95, STICK2=6.82, STICK3=3.91'
           '@SCALE VISC1=10.13,VISC2=5.6  , VISC3=10.9 '
```

```
          '@SCALE ONE1=5.95, TWO1=6.82, THREE1=3.91 '
          '@SCALE ONE2=5.95, TWO2=6.82, THREE2=3.91 '
          '@SCALE TORQ1=85, TORQ2=100, TORQ3 = 41 '
          '@PARAMETER INIQD1,INIQD2,INIQD3,INITQ1,INITQ2'
          '@PARAMETER INITQ3'
          '@MAXVAL INIQD1=2.25, INIQD2=1.6 , INIQD3=3.1, D22=6.37'
          '@MAXVAL INITQ1=2.8 , INITQ2=3.93, INITQ3=3.2 , D11=6.12'
          '@MINVAL INIQD1=-2.25, INIQD2=-1.6, INIQD3=-3.1, D22=4.0'
          '@MINVAL INITQ1=-2.8, INITQ2=-3.2, INITQ3=-3.93, D11=.5'
INITIAL
     '@BETA(BETA)'
     MAXT = PERIOD/BETA
     LOGPER = CINT * BETA
    '    '
    '   SET RUN CONTROL VARIABLES AND DEFINE VARIABLE TYPES  '
    '    '
    VARIABLE TIME = 0
     CONSTANT BETA =1, RUNTIM = 1.48, PERIOD =0.01401
     CONSTANT T1MAX=73, T2MAX=90, T3MAX=36, POINTR=.014
     CONSTANT CINT=.014, KV1=70.60, KV2=152.9, KV3=25.00
     CONSTANT KP1=563.4, KP2=1172, KP3=215,A=2.3562,B=0
     CONSTANT STATF1=5.95, STATF2=6.82, STATF3=3.91
     CONSTANT ICT1=5.95,ICT2=43,ICT3=-3.7
     CONSTANT ICSF1=5.95,ICSF2=6.82,ICSF3=-3.91
     INTEGER NUM
     '@PARAMETER BETA,  RUNTIM, POINTR '
     '@MAXVAL BETA =100,  RUNTIM= 7, TIME=50, POINTR=.01401'
     '@MINVAL BETA =.1, RUNTIM=0, TIME=0,POINTR=.001'
     ARRAY QEDD(3,720),QED(3,720),QE(3,720)
     REAL D011,D022,D033,C22,S2233,C2233,S33,C33,...
          T01,T02,T03,PE1,PE2,PE3,D012,D013,D023, ...
          GG2,GG3,VF1,VF2,VF3,STF1,STF2,STF3
     NSTEPS NSTP = 1
     LOGICAL ST1, ST2, ST3
     ' '
     '  LOAD THE DESIRED JOINT POSITIONS, VELOCITIES, AND  '
     '  ACCELERATIONS.  THESE ARE IN FILES GENERATED BY A   '
     '  SEPERATE PROGRAM CALLED S.TRAJEC  .                 '
     '   '
     CALL LOADING(QEDD,QED,QE,POINTR,A,B)
    ' '
      INIQD1 = 0
      INIQD2 = 0
      INIQD3 = 0
      INITQ1 = B
      INITQ2 = -A
      INITQ3 = A
      VE1=0.0
      VE2=0.0
      VE3=0.0
      PE1=0.0
      PE2=0.0
      PE3=0.0
  '  '
  '      INITIAL TORQUE IS INPUT INTO THE ARM TO ACCOUNT FOR '
  '  GRAVITY AND OTHER TERMS THAT EXIST PRIOR TO T=0.  '
  '  VARIABLES ICT1,ICT2,ICT3 ARE USED SO THAT THIS INITIAL '
  '  TORQUE CAN BE MODIFIED IN SIMRUN TO ACCOUNT FOR DIFFERENT '
  '  INITIAL CONDITIONS.  '
  '  '
      T01= ICT1
      T02= ICT2
      T03= ICT3
      QED(1,0)=0
      QED(2,0)=0
      QED(3,0)=0
      QE(1,B)=B
      QE(2,0)= - A
      QE(3,0)= A
      NUM = 0
      D011 = 0
      D022 = 0
      D033 = 0
      D012 = 0
      D013 = 0
      D023 = 0
      GG2 = 0
      GG3 = 0
  '   '
  '      INITIAL FRICTION VALUES ARE BASED ON INITIAL CONDITIONS '
  '  OF THE ARM.   '
  '   '
      STF1 = ICSF1
      STF2 = ICSF2
      STF3 = ICSF3
    END $'INITIAL'
DYNAMIC
   'Interrupt Rate Error Declarations'
       LOGICAL ENDER1,RATER1,ERROR1
       ENDER1 = .FALSE.
       ERROR1 = RATER1
DERIVATIVE
   '   '
  ' PROPORTIONAL PLUS DIRIVATIVE CONTROL LAW    '
  ' I.E. THIS IS THE CONTROLLER  '
  '   '
  '  VARIABLES -                                          '
  '       T01,T02,T03 - JOINT TORQUES                     '
  '       QED(I,J) - DESIRED JOINT VELOCITIES                    '
  '       QDA1,QDA2,QDA3 - ACTUAL JOINT VELOCITIES        '
  '       QE(I,J) - DESIRED JOINT TRAJECTORIES              '
  '       QA1,QA2,QA3 - ACTUAL JOINT TRAJECTORIES         '
  '       KV,KP - COEFFICIENTS USED TO POSITION THE       '
  '               CONTROLLERS POLES                      '
  '       VE1,VE2,VE3 - VELOCITY ERROR   '
  '       PE1,PE2,PE3 - POSITION ERROR   '
  '  '
    CALL INCRE(NUM,TIIME)
```

```
        C22 = COS(QA2)
        S22 = SIN(QA2)
        S2233 = SIN(QA2 + QA3)
        C2233 = COS(QA2 + QA3)
        S33 = SIN(QA3)
        C33 = COS(QA3)
        D011=2.4975 + 2.1007*C22**2 + 0.5323*S2233 - 0.033*C22*C2233 ...
               - 0.0405*C2233*S2233+ 0.9161*C22*S2233
        D022 = 5.419 + 0.9161*S33 - 0.0331*C33
        D033 =  1.1295
        D013 = -0.007*S2233 - 0.1596*C2233
        D012 = 2.4492 *S22 + D013
        D023 = 0.5468 + 0.4581*S33 - 0.0165*C33
        GG3 = 0.3761*C2233 - 10.4068*S2233
        GG2 = -52.106*C22 + 1.0972*S22 + GG3
        CALL STATIC(QDA1,QDA2,QDA3,T01,T02,T03,STF1,STF2,STF3)
        VE1 = QED(1,NUM) - QDA1
        VE2 = QED(2,NUM) - QDA2
        VE3 = QED(3,NUM) - QDA3
        PE1 = QE(1,NUM) - QA1
        PE2 = QE(2,NUM) - QA2
        PE3 = QE(3,NUM) - QA3
        T01 =   KV1 * VE1 + KP1 * PE1 + STF1
        T02 =   KV2 * VE2 + KP2 * PE2 + STF2 + GG2
        T03 =   KV3 * VE3 + KP3 * PE3 + STF3 + GG3
       '   '
      '     THIS SECTION CONTROLS A/D CONVERSIONS.  '
      '    '
        QA1 = Q1
        QA2 = Q2
        QA3 = Q3
        QDA1 = QD1
        QDA2 = QD2
        QDA3 = QD3
        TIIME=TIME
       '     '
       '     '
       '     '
    '@PARALLEL'
        '      THIS REGION CONTAINS THE ANALOG MODEL.    '
        '   '
        '   '
       '    REDUCING COMPUTATIONAL LOADING BY PRODUCING VARIABLES  '
    '  THAT ARE USED MORE THAN ONCE IN THE PARALLEL REGION.  THIS '
    '  MINIMIZES THE NUMBER OF SUMMERS AND MULTIPLIERS NEEDED TO '
    '  RUN THE MODEL.    '
    '    '
      C2 = COS(Q2)
      S2 = SIN(Q2)
      C3 = COS(Q3)
      S3 = SIN(Q3)
      QX = Q2 + Q3
      C23 = COS(QX)
      S23 = SIN(QX)
      C2S2 = C2*S2
      S2S23 = S2*S23
      C2S23 =  C2*S23
      C2C23 = C2*C23
      QD23 = QD2*QD3
      QD33 = QD3*QD3
      ''
    '  TORQUE VALUES COMPUTED FROM DERIVATIVE REGION '
      ''
      '  VARIABLES:                                                '
      '  '
      '     T1,T2,T3 - ANALOG VARIABLES OF JOINT TORQUES          '
      '     T01,T02,T03 - DIGITAL VARIABLES OF JOINT TORQUES      '
      '     '
      T1 = T01
      T2 = T02
      T3 = T03
      ''
    '  CALCULATING MODEL DYNAMIC'S COEFFICIENTS '
      ''
      '     VARIABLES:                                           '
      '     '
      '     D"IJ" - THE ROW "I", COLUMN "J" COMPONENT OF THE    '
      '             INERTIA MATRIX                              '
      '     D"IJK" - THE ROW "I", COLUMN "J" , DEPTH "K" COMPONENT '
      '              OF THE THIRD ORDER CORIOLIS AND CETRIFUGAL TENSOR'
      '     G1,G2,G3 - GRAVITY COMPONENTS ... G1 = 0                 '
      '     '
      D11 = 2.4975 + 2.1007*C2**2 + 0.5323*S23**2 ...
            + 0.9161*C2S23
      D22 = 5.419 + 0.9161*S3 - 0.0331*C3
      D12 = 2.4492*S2 + D13
      D13 = -0.007*S23 - 0.1596*C23
      D23 = 0.5468 + 0.4581*S3
      D11X = 0.5322*C3*S3 - 1.0643*S3*S2S23 + 0.4581*C2C23
      D112 = ( D11X - 1.5685*C2S2 - 0.4519*S2S23)
      D113 = ( D11X + 0.5322*C2S2)
      D122 = 1.9686*C2 + D123
      D123 = ( 0.1596*S23 - 0.007*C23)
      D133 = D123
      D211 = - D112
      D223 = ( 0.4581*C3 + 0.0165*S3)
      D233 = D223
      D311 = - D113
      D322 = - D223
      PROCEDURAL ( G2,G3 = C23,S23,C2,S2)
      '@IMPL (G3)'
      G2 = -52.106*C2 + 1.0972*S2 + G3
      G3 = 0.3761*C23 - 10.4068*S23
      '@END IMPL'
      END
      '    '
      '       THIS SECTION CALCULATES THE STATIC FRICTION OF EACH  '
      ' JOINT.  THE FRICTION IS A CONSTANT VALUE WHOSE SIGN IS '
```

```
' BASED ON THE SIGN OF THE JOINT VELOCITY.  IF JOINT  '
' VELOCITY IS BELOW A CERTAIN VALUE, THE SIGN OF THE   '
' FRICTION CONSTANT IS BASED ON THE DIRECTION OF APPLIED '
' TORQUE.   '
'   '
'  VARIABLES:   '
'      ST1,ST2,ST3 - LOGICAL VARIABLES USED TO DETERMINE '
'                 WHETHER TO USE VELOCITY OR TORQUE SIGN  '
'      ONE1,TWO1,THREE1 - PUTS OUT A (+/-)FRICTION BASED '
'                 ON DIRECTION OF TORQUE    '
'      ONE2,TWO2,THREE2 - PUTS OUT A (+/-)FRICTION BASED '
'                         ON DIRECTION OF VELOCITY    '
'      STICK1,STICK2,STICK3 - CHOSES PROPER FRICTION VALUE '
'                         BASED ON ST1,ST2,ST3    '
'   '
ST1 = ABS(QD1) .GT. 0.01
ST2 = ABS(QD2) .GT. 0.01
ST3 = ABS(QD3) .GT. 0.01
ONE1 = FCNSW(T1,-STATF1,0.0,STATF1)
ONE2 = FCNSW(QD1,-STATF1,0.0,STATF1)
TWO1 = FCNSW(T2,-STATF2,0.0,STATF2)
TWO2 = FCNSW(QD2,-STATF2,0.0,STATF2)
THREE1 = FCNSW(T3,-STATF3,0.0,STATF3)
THREE2 = FCNSW(QD3,-STATF3,0.0,STATF3)
STICK1 = RSW(ST1,ONE2,ONE1)
STICK2 = RSW(ST2,TWO2,TWO1)
STICK3 = RSW(ST3,THREE2,THREE1)
'   '
'    VISCOUS FRICTION    '
 VISC1 = 4.5*QD1
 VISC2 = 3.5*QD2
 VISC3 = 3.5*QD3
'   '
'    ADDITIVE TERMS IN THE MODEL.  THIS AVIODS A CODING PROBLEM'
'  ENCOUNTERED IN THE '@IMPL' REGION.  P-TRAN CONSIDERS ALL OF '
'  THE EQUATIONS IN THAT REGION AS ONE.  THERE ARE TOO MANY   '
'  VARIABLES ON THE RIGHT HAND SIDE IN THAT SECTION.   '
'   '
TORQ1 = - T1 + STICK1 + VISC1
TORQ2 = - T2 + STICK2 + G2 + VISC2 + D211*QD1*QD1
TORQ3 = - T3 + STICK3 + G3 + VISC3 + D311*QD1*QD1 ...
       +D322*QD2*QD2
''
' MODEL DYNAMIC EQUATIONS '
''
'  VARIABLES:                                              '
'  '
'  QDD1,QDD2,QDD3 - JOINT ACCELERATIONS                    '
'  QD1,QD2,QD3 -  JOINT VELOCITIES                         '
'  Q1,Q2,Q3 -  JOINT POSITIONS                             '
'  '
'      THIS NEXT SECTION CONTAINS AN ARITHMATIC LOOP CAUSED '
'  BY THE COUPLED NATURE OF THE MODEL.  THE PROCEDURAL AND  '
'   IMPL ARE NECESSARY TO INSTRUCT P-TRAN IN HANDLING THE   '
```

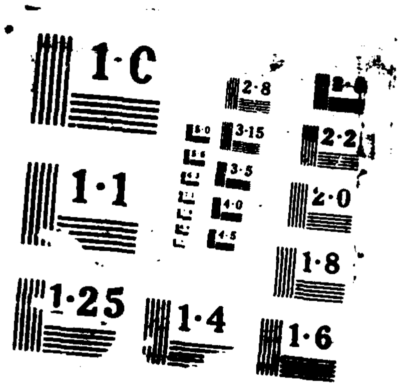
```
    '  SITUATION.   '
    '  '
    '  '
    PROCEDURAL (QDD1,QDD2,QDD3  = D12,D13,D122,QD2 ...
         ,D123,QD23,D133,QD33,D11,D23,D223 ...
         ,D233,QD33,D22,TORQ1,TORQ2,TORQ3)
    '@IMPL (QDD3,QDD2)'
    QDD1 = -( TORQ1 + D12*QDD2 + D13*QDD3 + D122*QD2*QD2 ...
        + 2*D123*QD23 + D133*QD33 + 2*D112*QD1*QD2 ...
        + 2*D113*QD1*QD2)/D11
    QDD2 = -( TORQ2 + D23*QDD3 + 2*D223*QD23 + D233*QD33 ...
                + D12 * QDD1)/D22
    QDD3 = -0.88535 * ( TORQ3  + D13 * QDD1 ...
         + D23 * QDD2)
    '@END IMPL'
    END
    QD1  = INTEG(QDD1,INIQD1)
    QD2  = INTEG(QDD2,INIQD2)
    QD3  = INTEG(QDD3,INIQD3)
    Q1   = INTEG(QD1,INITQ1)
    Q2   = INTEG(QD2,INITQ2)
    Q3   = INTEG(QD3,INITQ3)
        TERMT(TIME .GT. RUNTIM)
    '    '
    'DEFINE INTERRUPT CONTROL'
    '  '
       LOGICAL GPI0,GPI1
       GPI0 = CLOCK(PERIOD)
       GPI1 = CLOCK(LOGPER)
       '@INTRRT 1 =GPI0'
       '@INTRRT 2 =GPI1'
        RATER1 = RATERR(GPI0,ENDER1)
       '@RECORD(REC01,,,,,,,,,,,)'
       '@ENDPARALLEL'
  END $ 'OF DERIVATIVE'
  END $ 'OF DYNAMIC'
  TERMINAL $ END $ 'OF TERMINAL'
  END $ 'OF PROGRAM'
  *TRANSLATE
    '  '
    '  SET UP A/D AND D/A CONVERTERS  '
    '  '
      DCA(1) = T01,T02,T03
      PADC(1) = Q1,Q2,Q3,QD1,QD2,QD3,TIME
  *OUTPUT
  *END
            SUBROUTINE PREP1
  +
        INCLUDE E1.PDG
           Q1 = QRPADC(0)*S:Q1
           Q2 = QRPADC(1)*S:Q2
           Q3 = QRPADC(2)*S:Q3
           QD1 = QRPADC(3)*S:QD1
           QD2 = QRPADC(4)*S:QD2
```

```
            QD3 = QRPADC(5)*S:QD3
           TIME = QRPADC(6)*S:TIME
            RETURN
            END
C
            SUBROUTINE POST1
+
         INCLUDE E1.PDG
            COMMON /QQDCP/DCASF(0:2)
            LOGICAL DELAY
            CALL QWDCAR(0,T01*DCASF(0))
            CALL QWDCAR(1,T02*DCASF(1))
            CALL QWDCAR(2,T03*DCASF(2))
            IF (L:RATER1) CALL ZZRTER(1)
            L:ENDER1 = .TRUE.
            DELAY = L:ENDER1
            L:ENDER1 = .FALSE.
            RETURN
            END
C
            SUBROUTINE PREPDCA
+
            COMMON /QQDCP/DCASF(0:2)
            DCASF(0) = 1.0/QDCASR(0)/T1MAX
            DCASF(1) = 1.0/QDCASR(1)/T2MAX
            DCASF(2) = 1.0/QDCASR(2)/T3MAX
            RETURN
            END
C
            SUBROUTINE LOADING(QEDD,QED,QE,POINTER,A,B)
            DIMENSION QEDD(3,720),QED(3,720),QE(3,720)
            REAL TF,POINTER,INPOS1,INPOS2,INPOS3
            INTEGER NUMPTS,STAT1,STAT2,STAT3
            TF=1.5
            INPOS1 = B
            INPOS2 = -A
            INPOS3 = A
            NUMPTS=IFIX(TF/0.007) + 1
            OPEN(UNIT=13,
     1      OPENMODE='R',BLOCKED=.TRUE.)
            OPEN(UNIT=11,
     1      OPENMODE='R',BLOCKED=.TRUE.)
            OPEN(UNIT=12,
     1      OPENMODE='R',BLOCKED=.TRUE.)
            DO 300 J=1,NUMPTS
            READ(13,400) (QE(I,J),I=1,3)
            READ(11,400) (QED(I,J),I=1,3)
            READ(12,400) (QEDD(I,J),I=1,3)
            WRITE(6,46) QE(1,J),QE(2,J),QE(3,J)
            WRITE(6,47) QED(1,J),QED(2,J),QED(3,J)
   46       FORMAT ('QE',2X,3(E12.6))
   47       FORMAT ('QED',30X,3(E12.6))
            QE(1,J) = QE(1,J) + INPOS1
            QE(2,J) = QE(2,J) + INPOS2
```

```
          QE(3,J) = QE(3,J) + INPOS3
          QED(3,J) = QED(3,J)
          QEDD(3,J) = QEDD(3,J)
 300      CONTINUE
 400      FORMAT(3(E12.6))
          CLOSE(UNIT=13,STATUS='KEEP')
          CLOSE(UNIT=11,STATUS='KEEP')
          CLOSE(UNIT=12,STATUS='KEEP')
          RETURN
          END
C
        SUBROUTINE STATIC(QDA1,QDA2,QDA3,T01,T02,T03,STF1,STF2,STF3)
          REAL QDA1,QDA2,QDA3,T01,T02,T03,STF1,STF2,STF3
             IF (ABS(QDA1) .GT. 0.01) THEN
                    STF1 = SIGN(5.95,QDA1)
             ELSE
                    STF1 = SIGN(5.95,T01)
             ENDIF
             IF (ABS(QDA2) .GT. 0.01) THEN
                    STF2 = SIGN(6.82,QDA2)
             ELSE
                    STF2 = SIGN(6.82,T02)
             ENDIF
             IF (ABS(QDA3) .GT. 0.01) THEN
                    STF3 = SIGN(3.91,QDA3)
             ELSE
                    STF3 = SIGN(3.91,T03)
             ENDIF
          RETURN
          END
C
         SUBROUTINE INCRE(DNUM,DTIIME)
         INTEGER DNUM,STEP
         REAL DTIIME
         STEP = 2
         IF (DTIIME .LT. 0.021) THEN
              DNUM = 1
         ELSE
              DNUM = DNUM + STEP
         ENDIF
         RETURN
         END
```

AD-A193 119 DEVELOPMENT OF A HYBRID SIMULATOR FOR ROBOTIC MANIPULATORS(U) AIR FORCE INST OF TECH WRIGHT-PATTERSON AFB OH SCHOOL OF ENGINEERING P M VAN WIRT DEC 87 2/2

UNCLASSIFIED AFIT/GE/ENG/87D-68 F/G 12/5 NL


1·0
2·8
2·5
3·15
5·0
2·2
3·5
1·1
4·0
2·0
4·5
1·8
1·25
1·4
1·6

Appendix B

Trajectory Information

This program generates the origional trajectory used to debug the simulation. It suffers from some severe restrictions because it produces actuator saturation due to violation of jerk constraints.

```
PROGRAM
  '     '
  '         THIS PROGRAM GENERATES THE POSITION, VELOCITY, AND  '
  '     ACCELERATIONS FOR A PUMA 560 ROBOT ARM"S JOINTS.  IT DOES '
  '     THIS BASED ON A POLYNOMIAL EXPRESSION OF A TRAJECTORY.    '
   '     '
     ARRAY QDSI(6,720),QDST(6,720),QDSTT(6,720),Q0(6),QF(6),A(6)
     REAL      DELT,TF,TIME,B,C,XYZ,APVW,BTMC,BTMCS
     INTEGER NUMPTS
     LOGICAL STOP
INITIAL
         ''
'   INITIALIZE PARAMETERS  '
          ''
   '  VARIABLES:   '
   '      Q0(I) - INITIAL JOINT POSITION          '
   '      QF(I) - FINAL JOINT POSITION            '
   '       '
     STOP = .TRUE.
       Q0(0) = 0.0
      Q0(1) = 0.0
      Q0(2) = 0.0
      Q0(3) =0.0
      Q0(4) = 0.0
      Q0(5) =0.0
      QF(0) = 1.133
      QF(1) = .755
         QF(2) = 1.51
         QF(3) = 1.7
         QF(4) = 1.7
        XYZ = 0.63662
         QF(5) = 1.7
       ''
'  DEFINE TOTAL TIME,SAMPLING TIME AND CURVE SHAPE '
     ''
     '     VARIABLES:   '
     '       DELT - SAMPLING TIME           '
     '       TF - FINAL TIME                '
     '       B -        '
     '       C -        '
     '  '
         DELT=0.007
```

```
          TF= 1.5
          B=6
          C=4.54
END $ 'OF INITIAL'
DYNAMIC
DERIVATIVE
        TERMT(STOP .OR. (T .GT. 0.1))
     ''
'  CALCULATE TRAJECTORIES   '
    ''
    '   VARIABLES:      '
    ''
    '    A(I) - SETS UP EACH JOINT GENERATION BASED ON INITIAL AND '
    '           FINAL VALUES      '
    '    NUMPTS - NUMBER OF INTERMEDIATE STEPS       '
    '    QDSI -  JOINT POSITION      '
    '    QDST -  JOINT VELOCITY       '
    '    QDSTT - JOINT ACCELERATION     '
    '  '
         DO 70 I=0,5
         A(I)=(QF(I)-Q0(I))/(1.0 + (XYZ*(ATAN(C ))))
  70..  CONTINUE
         NUMPTS= IFIX(TF/DELT) + 1
        DO 100 J=1,NUMPTS
         TIME=FLOAT(J-1  )*DELT
         BTMC=B*TIME-C
         BTMCS=BTMC**2
         DO 100 K=0,5
          APVW=A(K)*XYZ
          QDSI(K,J)=Q0(K)+APVW*(ATAN(BTMC)+ATAN(C))
          QDST(K,J)=APVW*B/(1.0+BTMCS)
         QDSTT(K,J)= -2.0*APVW*(B**2)*BTMC/((1.0+BTMCS**2))
  100..  CONTINUE
         ''
'  STORE TRAJECTORIES  '
''
  CALL STORE(QDSI,QDST,QDSTT,NUMPTS)
        CONTINUE
END $ 'OF DERIVATIVE'
END $ 'OF DYNAMIC'
TERMINAL $ END $ 'OF TERMINAL'
END $ 'OF PROGRAM'
       SUBROUTINE STORE(QDSI,QDST,QDSTT,NUMPTS)
       DIMENSION QDSI(6,720),QDST(6,720),QDSTT(6,720)
         INTEGER NUMPTS
         OPEN(UNIT=10,FILE='S.PTRJ2',OPENMODE='W',BLOCKED=.TRUE.)
         OPEN(UNIT=11,FILE='S.VTRJ2',OPENMODE='W',BLOCKED=.TRUE.)
         OPEN(UNIT=12,FILE='S.ATRJ2',OPENMODE='W',BLOCKED=.TRUE.)
C
C  STORE DATA
C
         DO 300 J=1,NUMPTS
         WRITE(10,400) (QDSI(I,J),I=0,5)
         WRITE(11,400) (QDST(I,J),I=0,5)
```

```
          WRITE(12,400) (QDSTT(I,J),I=0,5)
  300     CONTINUE
          WRITE(10,410)
          WRITE(11,410)
          WRITE(12,410)
  400     FORMAT(1X,6 (E12.6))
  410     FORMAT(1X,//)
C
          CLOSE(UNIT=10,STATUS='KEEP')
          CLOSE(UNIT=11,STATUS='KEEP')
          CLOSE(UNIT=12,STATUS='KEEP')
          RETURN
          END
```

The following three plots show the trajectory used to validate the simulation. This trajectory was generated using cubic splines and avoids violation of actuator constraints. The trajectory data can be found in S1.PSPLA1 (position), S1.VSPLA1 (velocity), and S1.ASPLA1.

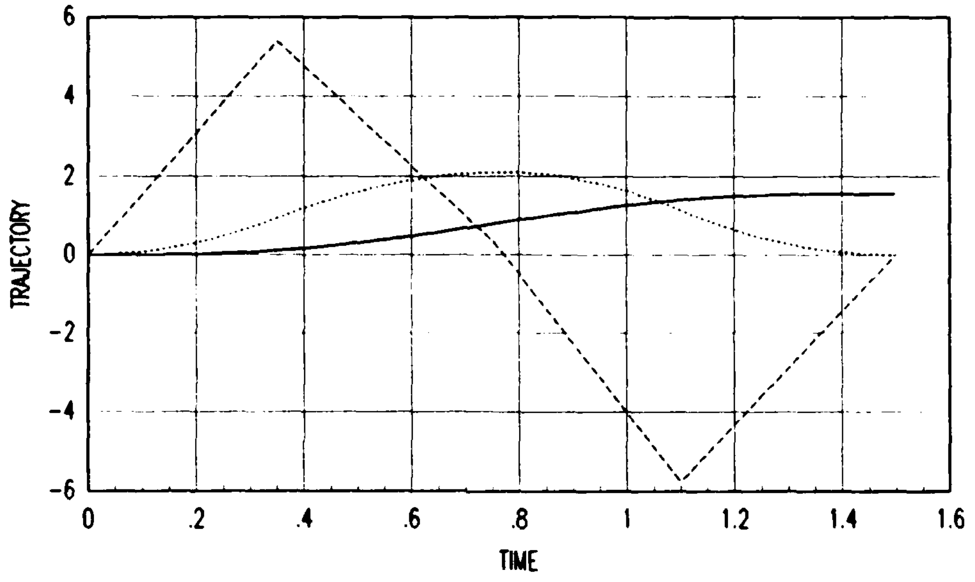


Figure C.1. Joint One Trajectory

Position	________
Velocity	....
Acceleration	-------

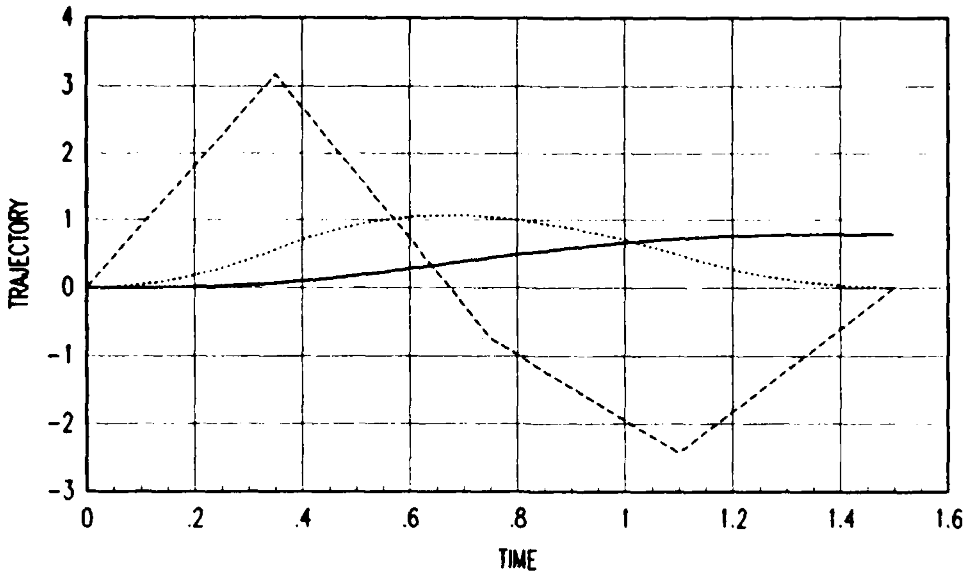


Figure C.2. Joint Two Trajectory

Position	________
Velocity	....
Acceleration	-------

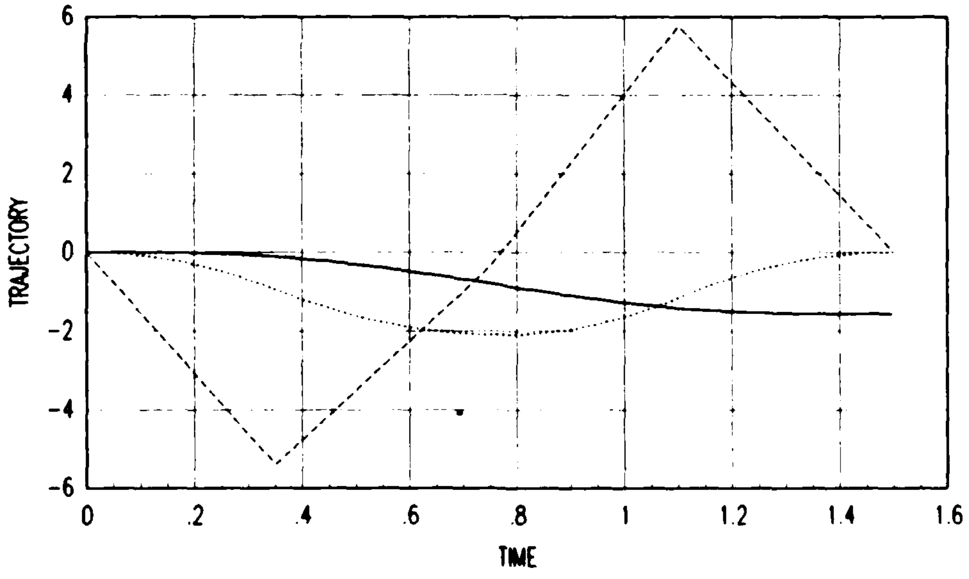


Figure C.3. Joint Three Trajectory

Position	________
Velocity	....
Acceleration	-------

Appendix C

MATRIXx Configuration Software

Index of Configuration Software

This program converts data from SIMSTAR format into MATRIXx format. The subroutine MATSAV is proprietary software provided by MATRIXx that converts a matrix into a file containing properly formatted MATRIXx data.

```
C        THIS PROGRAM TAKES ERROR DATA FROM THE ROBOT DATA FILES
C     IN SIMSTAR AND CONVERTS IT INTO THE MATRIXX FORMAT FOR USE
C     IN MATRIXX.
C
C         WRITTEN BY: CAPT PETER M VAN WIRT
C              22 SEPT 87
C         NO RIGHTS RESERVED
C
       PROGRAM CONSSMATX
       CHARACTER NAME,POSITION*10,DATAFL*12
       CHARACTER MATRDATA*12,TIMER*12,DUM*2
       DOUBLE PRECISION PER(110,3),DUMMY,TIME(110,1)
       INTEGER J,I,DUM2,K,L
C
       WRITE(6,10)
  10   FORMAT(2X,//,30X,'WELCOME TO CONSTMATX',//,2X,'OBJECTIVE:  ',
     1        'CONVERT SIMSTAR SIMULATION ERROR DATA TO MATRIXX'
     1        ,' FOR PLOTTING',/)
       WRITE(6,*) 'INPUT DATA FILE NAME'
       READ(5,40) DATAFL
  40   FORMAT(A12)
C
       WRITE(6,*) 'INPUT TIME VARIABLE NAME'
       READ(5,40) TIMER
C
       WRITE(6,*) 'INPUT MATRIXX DATA FILE NAME'
       READ(5,40) MATRDATA
C
```

```
C
      OPEN(UNIT=10,TYPE='OLD',NAME=DATAFL)
      OPEN(UNIT=11,TYPE='NEW',NAME=MATRDATA)
C
      WRITE(6,*) 'INPUT DESIRED DATA MATRIX NAME'
      READ(5,50) POSITION
 50   FORMAT(A8)
C
C         DEVELOPE TIME VECTOR AND PUT IN MATRIXX FORMAT
C
       DO 100 K=0,108
         L = K + 1
         TIME(L,1)=K* 0.014
 100   CONTINUE
          CALL MATSAV(11,TIMER,100,108,1,0,TIME,DUMMY,'(E10.4)')
C
C        READ DATA FROM FILE AND PUT IT IN ARRAY
C   THEN CALL SUBROUTINE THAT PUTS ARRAY IN
C   MATRIXX FORMAT.
C
          DO 200 I=1,108
              READ(10,500) DUM2,PER(I,1),PER(I,2),PER(I,3)
 200      CONTINUE
 500      FORMAT(3X,I2,3X,G9.7,3X,G9.7,3X,G9.7)
          CALL MATSAV(11,POSITION,100,108,3,0,PER,DUMMY,'(F9.7)')
      STOP
      END
```

This program coverts step test data from PUMA format into MATRIXx format.

```
C       THIS PROGRAM TAKES ERROR DATA FROM THE ROBOT DATA FILES
C     ,FROM STEP INPUT RUNS,
C     AND CONVERTS IT INTO THE MATRIXX FORMAT FOR USE IN MATRIXX.
C
C         WRITTEN BY: 1LT PETER M VAN WIRT
C              30 JUN 87
C         NO RIGHTS RESERVED
C
      PROGRAM STEPVER
      CHARACTER NAME,POSITION*10,DATAFL*12
      CHARACTER MATRDATA*12,TIMER*12,DUM*2
      DOUBLE PRECISION PER(100,2),DUMMY,TIME(100,1)
      INTEGER J,I,DUM2,K,L
C
      WRITE(6,10)
   10 FORMAT(2X,//,30X,'WELCOME TO CONSTMATX',//,2X,'OBJECTIVE:  ',
     1       'CONVERT RHCS STEP TEST RESPONSE DATA TO MATRIXX'
     1       ,' FOR PLOTTING',/)
C
      WRITE(6,*) 'INPUT DATA FILE NAME'
      READ(5,40) DATAFL
   40 FORMAT(A12)
C
      WRITE(6,*) 'INPUT TIME VARIABLE NAME'
      READ(5,40) TIMER
C
      WRITE(6,*) 'INPUT MATRIXX DATA FILE NAME'
      READ(5,40) MATRDATA
C
C
      OPEN(UNIT=10,TYPE='OLD',NAME=DATAFL)
      OPEN(UNIT=11,TYPE='NEW',NAME=MATRDATA)
C
      WRITE(6,*) 'INPUT DESIRED DATA MATRIX NAME'
      READ(5,50) POSITION
   50 FORMAT(A8)
C
C     STRIPS OFF TOP OF DATA FILE
C
      READ(10,60) DUM
   60 FORMAT(A1,//////)
C
C          DEVELOPE TIME VECTOR AND PUT IN MATRIXX FORMAT
C
       DO 100 K=0,71
         L = K + 1
         TIME(L,1)=K* 0.007
  100  CONTINUE
```

```
              CALL MATSAV(11,TIMER,100,71,1,0,TIME,DUMMY,'(E10.4)')
C
C          READ DATA FROM FILE AND PUT IT IN ARRAY
C     THEN CALL SUBROUTINE THAT PUTS ARRAY IN
C     MATRIXX FORMAT.
C
              DO 200 I=1,36
                  J = I + 36
                   READ(10,500) DUM2,PER(I,1),PER(I,2),DUM2,PER(J,1),
     1  PER(J,2)
  200         CONTINUE
  500         FORMAT(13X,I2,2X,F9.4,1X,F9.4,8X,I2,3X,F9.4,1X,F9.4,/)
              CALL MATSAV(11,POSITION,100,71,2,0,PER,DUMMY,'(F9.4)')
         STOP
         END
```

This program converts data from PUMA error files into MATRIXx.

```
C       THIS PROGRAM TAKES ERROR DATA FROM THE ROBOT DATA FILES
C     AND CONVERTS IT INTO THE MATRIXX FORMAT FOR USE IN MATRIXX.
C         WRITTEN BY: 1LT PETER M VAN WIRT
C              29 JUN 87
C         NO RIGHTS RESERVED
C
        PROGRAM CONTERMATX
        CHARACTER NAME,POSITION*10,VELOCITY*10,DATAFL*12
        CHARACTER MATRDATA*12,TIMER*12
        DOUBLE PRECISION X(300,6),V(300,6),DUMMY,SAMPLE,TIME(300,1)
        INTEGER N,J,I,DUM2,K,L
C
        WRITE(6,10)
   10   FORMAT(2X,//,30X,'WELCOME TO CONTERMATX',//,2X,'OBJECTIVE:  ',
     1        'CONVERT RHCS TRAJECTORY TRACKING ERROR DATA TO MATRIXX'
     1        ,' FOR PLOTTING',/)
        WRITE(6,*) 'INPUT DATA FILE NAME'
        READ(5,40) DATAFL
   40   FORMAT(A12)
C
        WRITE(6,*) 'INPUT DESIRED TIME VARIABLE NAME'
        READ(5,40) TIMER
C
        WRITE(6,*) 'INPUT DESIRED MATRIXX DATA FILE NAME'
        READ(5,40) MATRDATA
        OPEN(UNIT=10,TYPE='OLD',NAME=DATAFL)
        OPEN(UNIT=11,TYPE='NEW',NAME=MATRDATA)
C
        WRITE(6,*) 'INPUT DESIRED POSITION DATA NAME'
        READ(5,50) POSITION
        WRITE(6,*) 'INPUT DESIRED VELOCITY DATA NAME'
        READ(5,50) VELOCITY
   50   FORMAT(A8)
C
C       READ HEADER OFF OF DATA FILE
C
        READ(10,400) NAME,N,SAMPLE,DUM2
C
C           BUILD TIME VECTOR
C
         DO 100 K=1,N
           L = K - 1
           TIME(K,1)=L*SAMPLE
  100    CONTINUE
             CALL MATSAV(11,TIMER,300,N,1,0,TIME,DUMMY,'(E10.4)')
C
C           READ IN DATA AND PUT IT IN ARRAYS, THEN CALL
C      MATSAV WHICH PUTS THE ARRAYS IN MATRIXX FORMAT
C
             DO 200 J=1,N
                  READ(10,500) (X(J,I),I=1,6)
```

```
200        CONTINUE
400        FORMAT(1X,A7,2X,I3,2X,E10.4,2X,I3)
500        FORMAT(1X,6(E10.4))
           DO 600 J=1,N
                READ(10,500) (V(J,I),I=1,6)
600        CONTINUE
           CALL MATSAV(11,POSITION,300,N,6,0,X,DUMMY,'(E10.4)')
           CALL MATSAV(11,VELOCITY,300,N,6,0,V,DUMMY,'(E10.4)')
      STOP
      END
```

Appendix D

Simulation vs Actual Error Profiles

This appendix contains error profiles from initial condition zero.

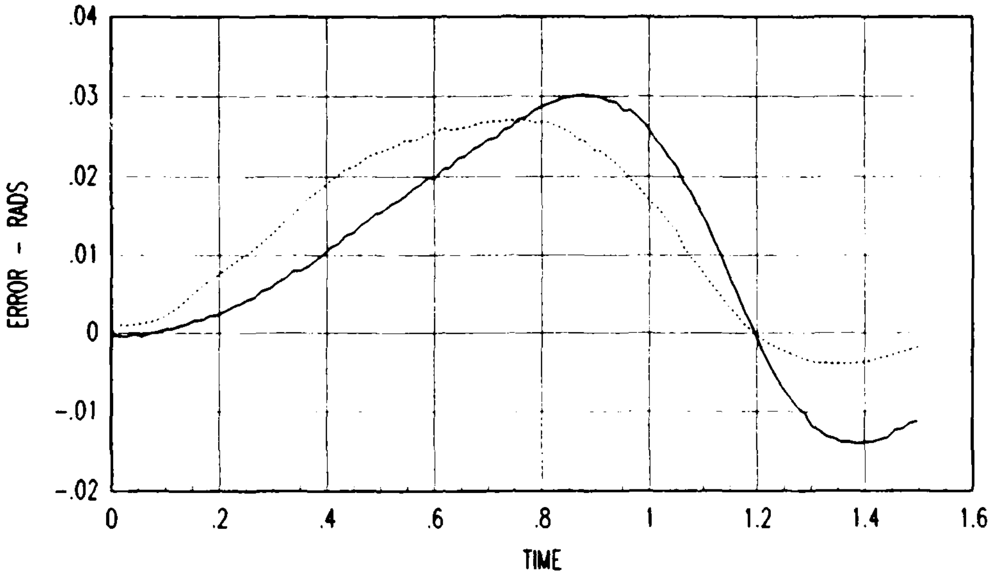


Figure D.1. Joint One PD Error Profile (I.C. 0)

Simulation ______
Actual ......

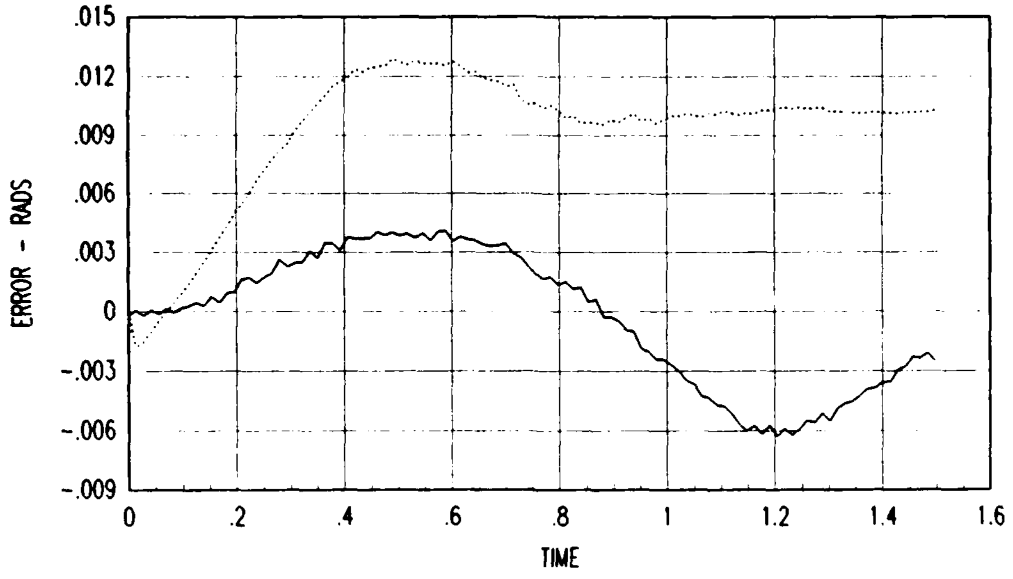


Figure D.2. Joint Two PD Error Profile (I.C. 0)

Simulation ______
Actual ......

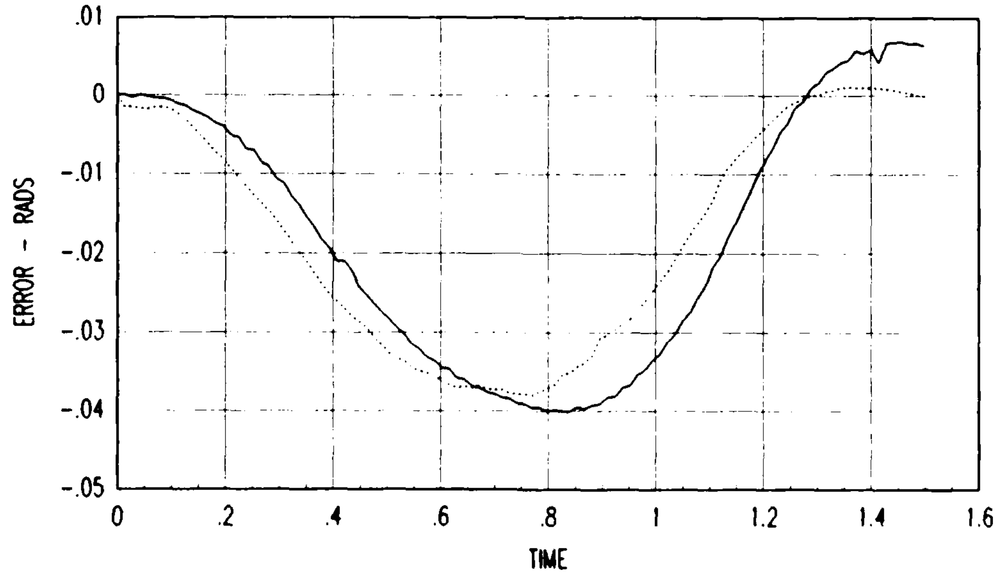


Figure D.3. Joint Three PD Error Profile (I.C. 0)

Simulation ______
Actual ......

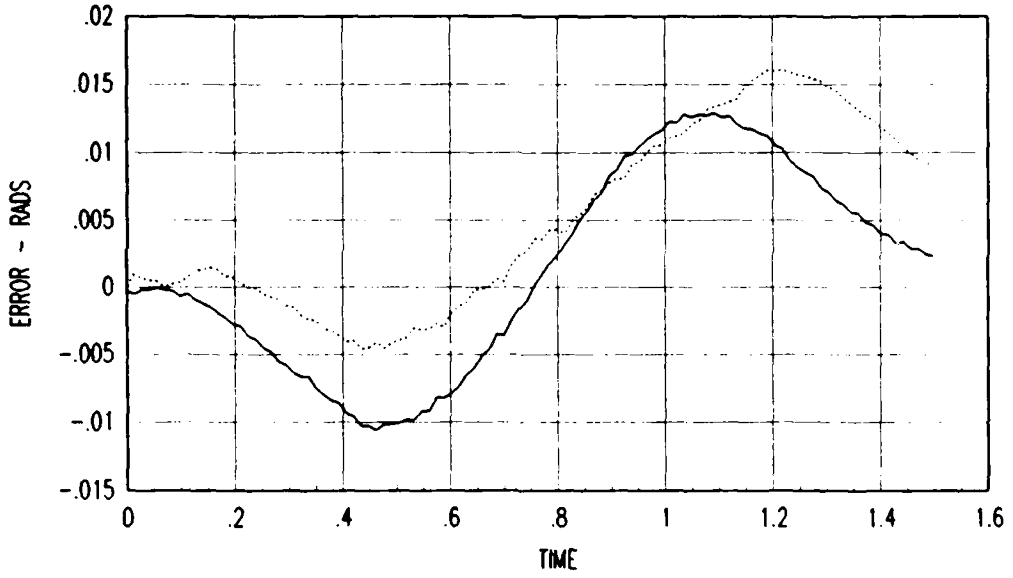


Figure D.4. Joint One Feedforward/Diagonal Error Profile (I.C.0)

Simulation ______
Actual ......

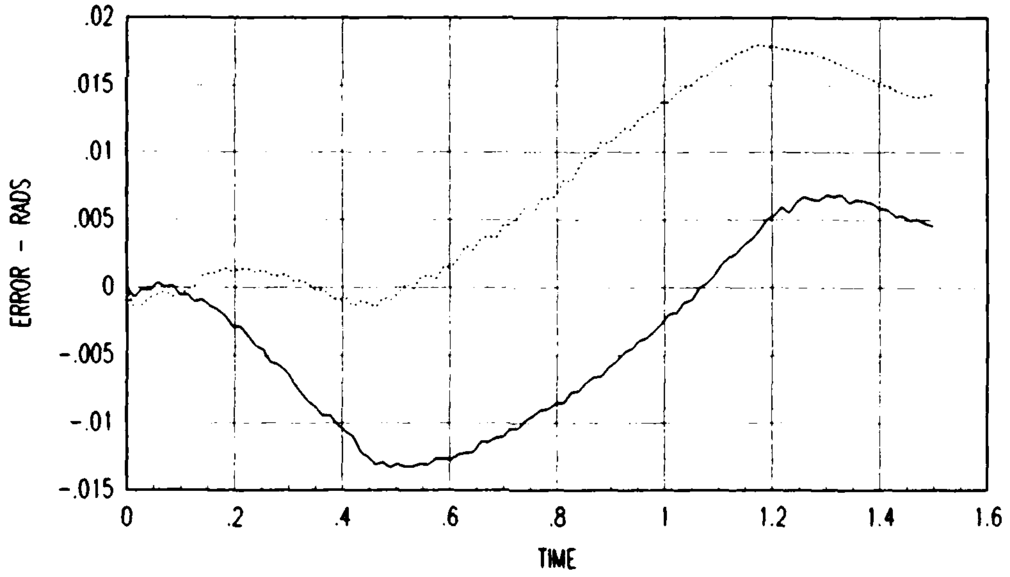


Figure D.5. Joint Two Feedforward/Diagonal Error Profile (I.C.0)

Simulation ______
Actual ......

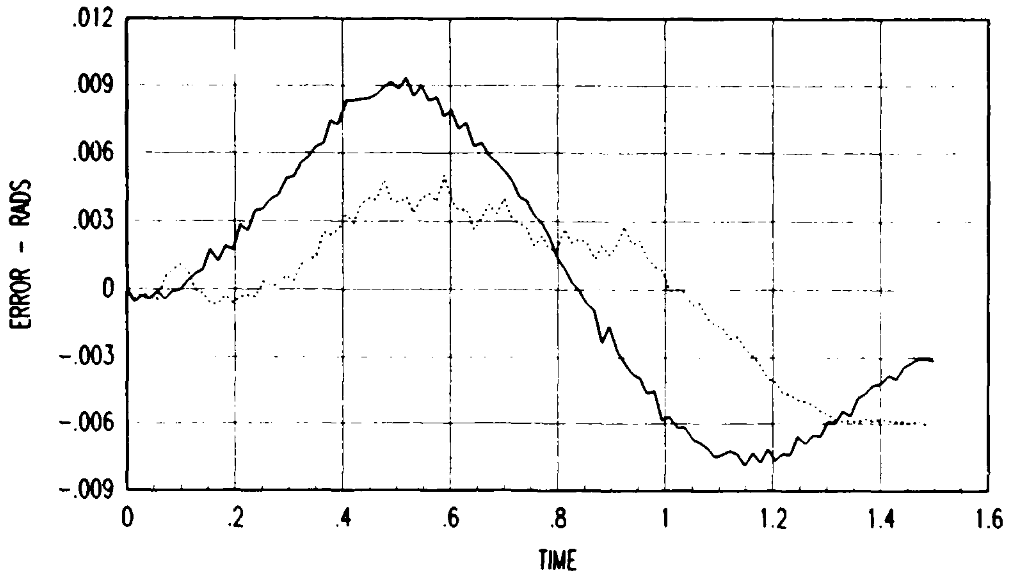


Figure D.6. Jt Three Feedforward/Diagonal Error Profile (I.C.0)

Simulation ______
Actual ......

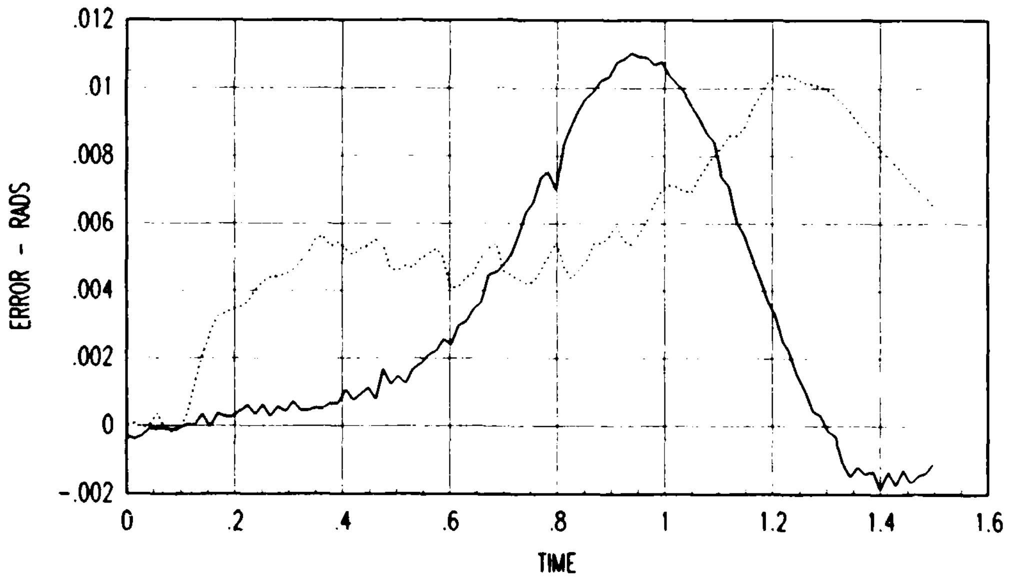


Figure D.7. Joint One Feedforward/Full Error Profile (I.C. 0)

Simulation ______
Actual ......

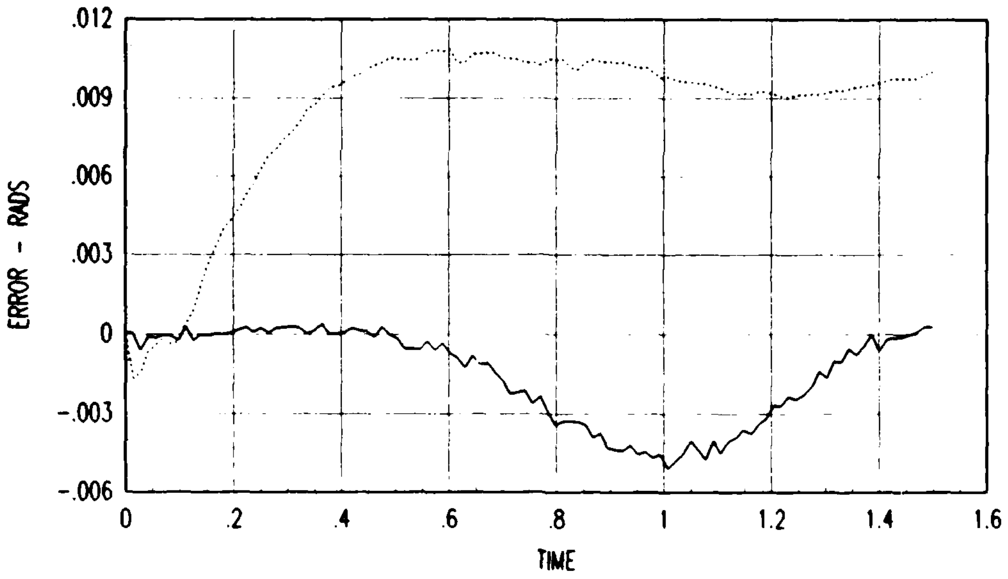


Figure D.8. Joint Two Feedforward/Full Error Profile

Simulation ______
Actual ......

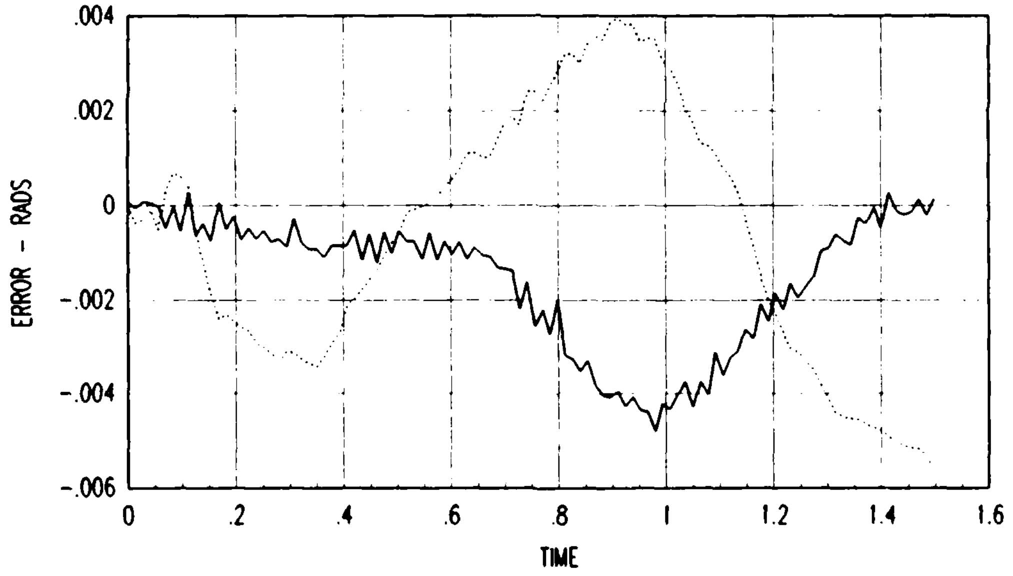


Figure D.9. Joint Three Feedforward/Full Error Profile (I.C. 0)

Simulation ______
Actual ......

Appendix E

Additional Step Test Results

This appendix contains additional step response plots used in determining the viscous friction coefficients.

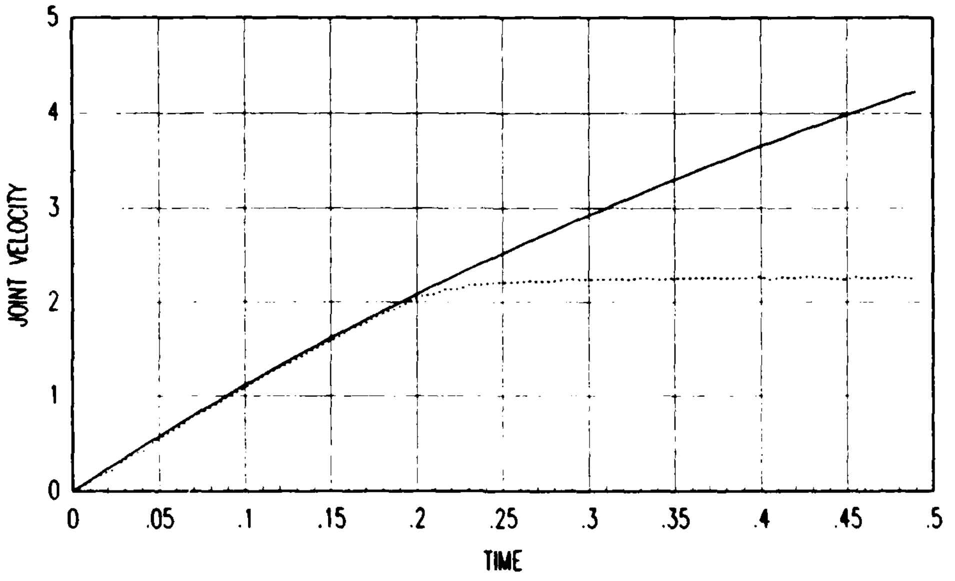


Figure E.1. Joint One Step Response, Counts = 750

Ideal Model ......
Actual Response ______

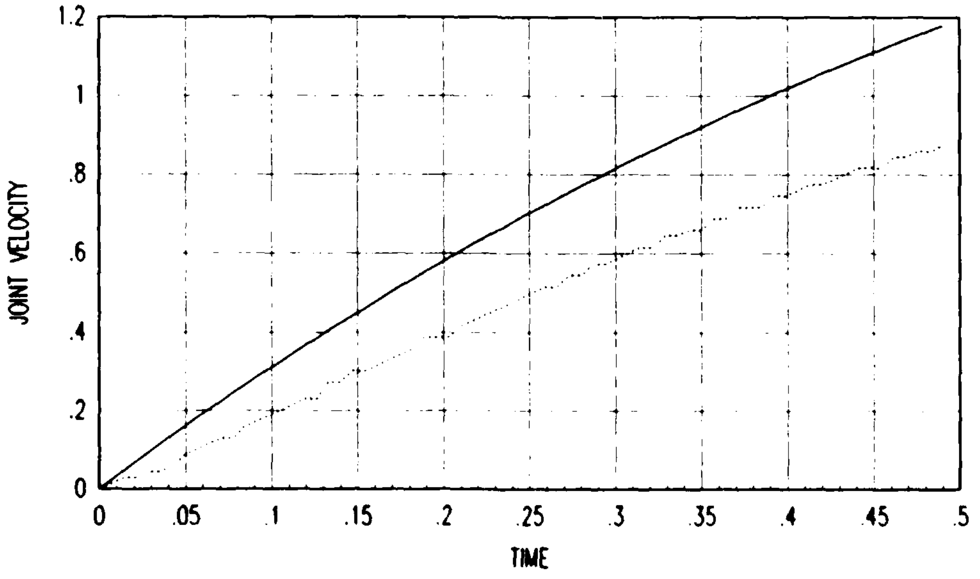


Figure E.2. Joint One Step Response, Counts = 300

Ideal Model ......
Actual Response ______

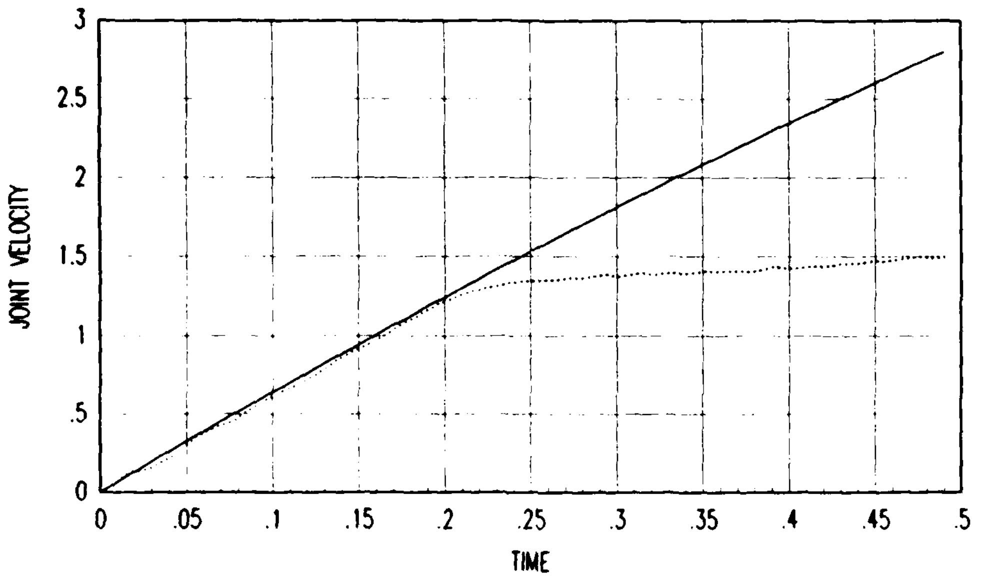


Figure E.4. Joint Two Step Response, Counts = 500

Ideal Model ......
Actual Response ______

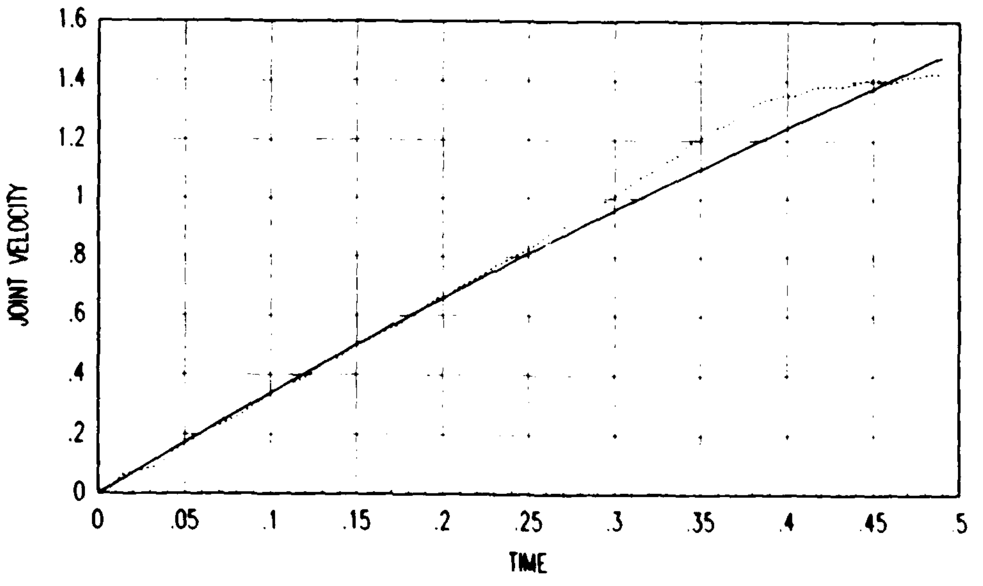


Figure E.5. Joint Two Step Response, Counts = 300

Ideal Model ......
Actual Response ______

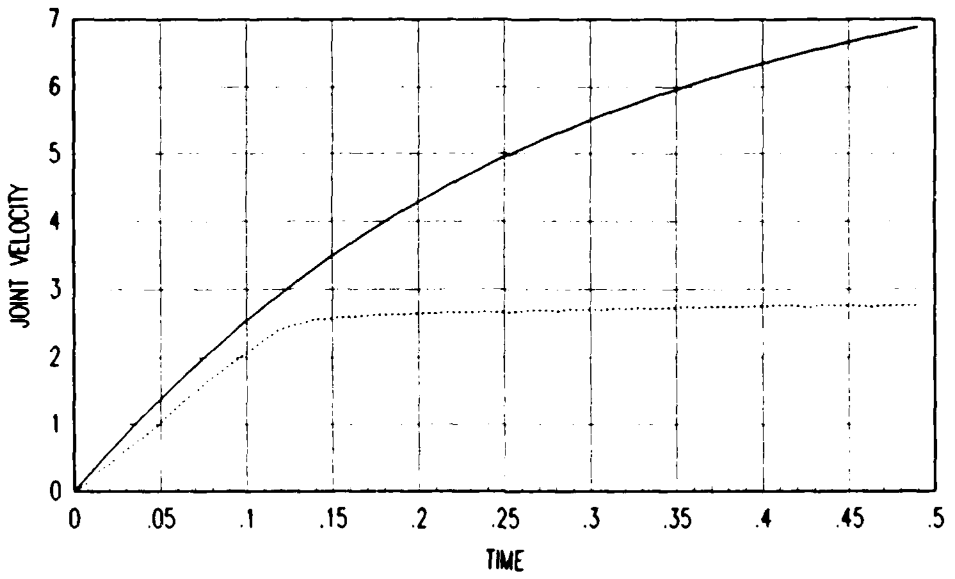


Figure E.7. Joint Three Step Response, Counts = 750

Ideal Model ......
Actual Response ______

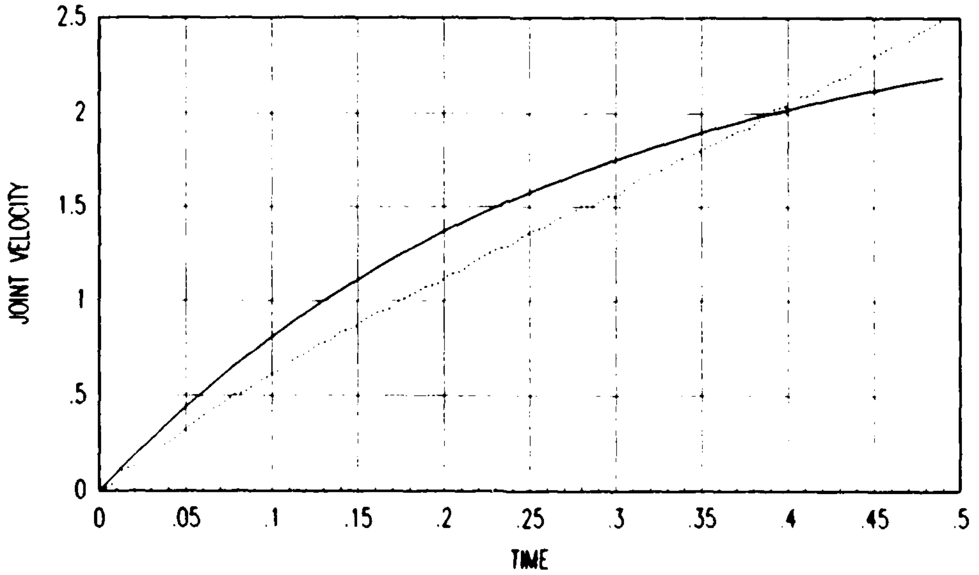


Figure E.8. Joint Three Step Response, Counts = 300

Ideal Model ......
Actual Response ______

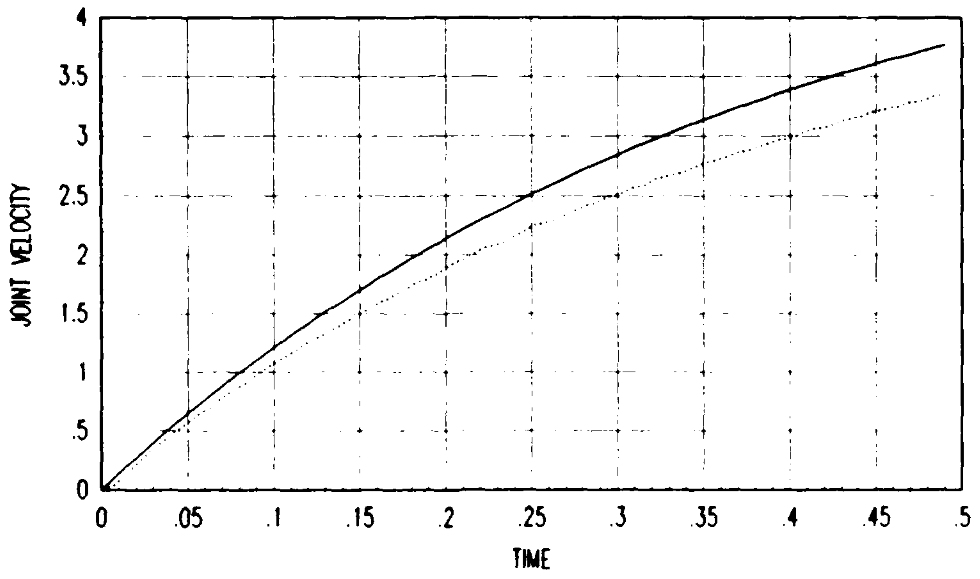


Figure E.9. Joint Four Step Response, Counts = 300

Ideal Model ......
Actual Response ______

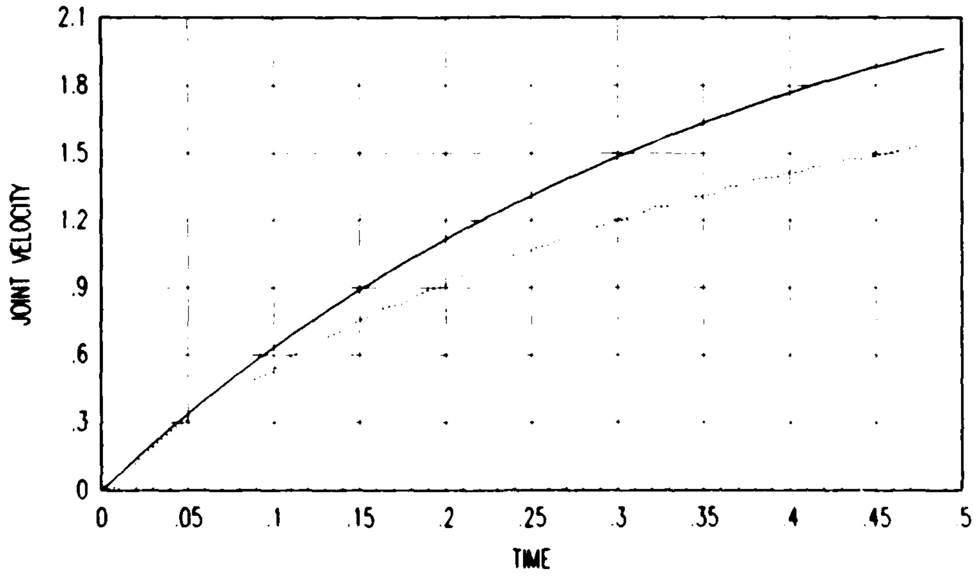


Figure E.10. Joint Four Step Response, Counts = 200

Ideal Model ......
Actual Response ______

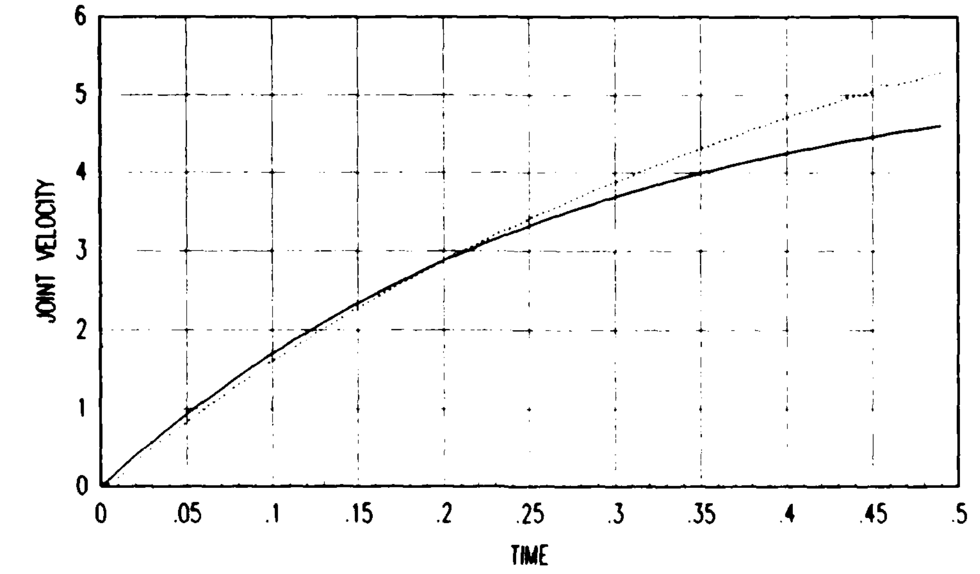


Figure E.11. Joint Five Step Response, Counts = 400

Ideal Model ......
Actual Response ______

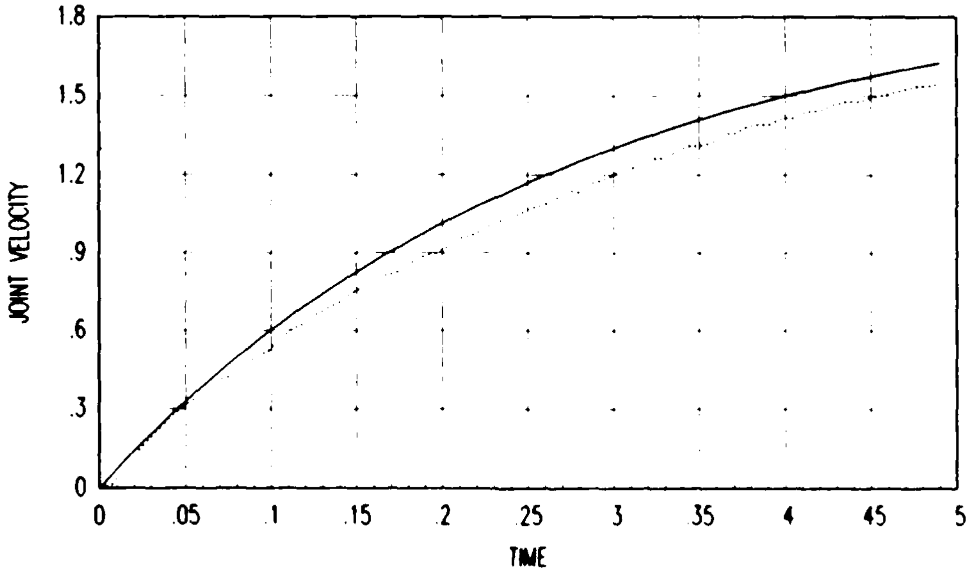


Figure E.12. Joint Five Step Response, Counts = 200

Ideal Model ......
Actual Response ______

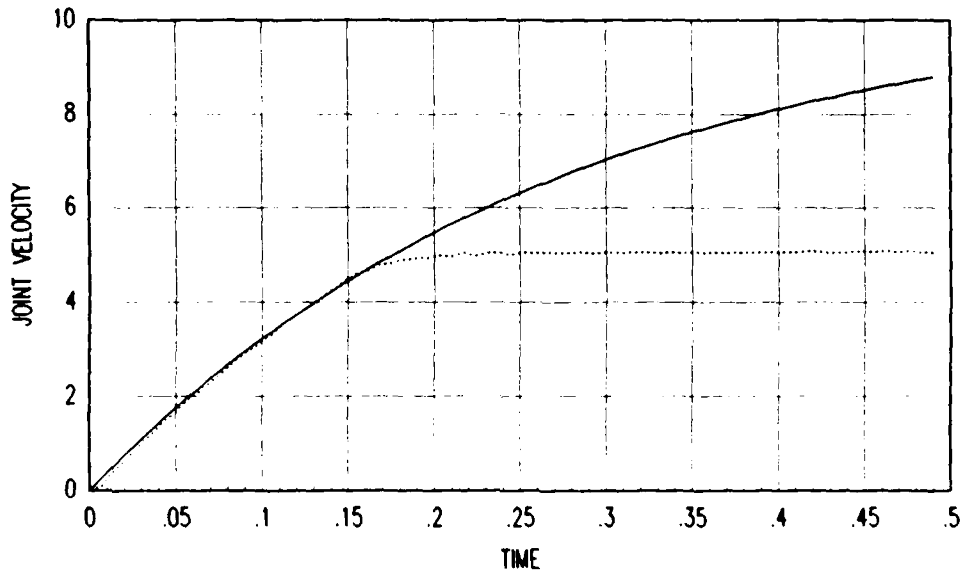


Figure E.13. Joint Six Step Response, Counts = 750

Ideal Model ......
Actual Response ______

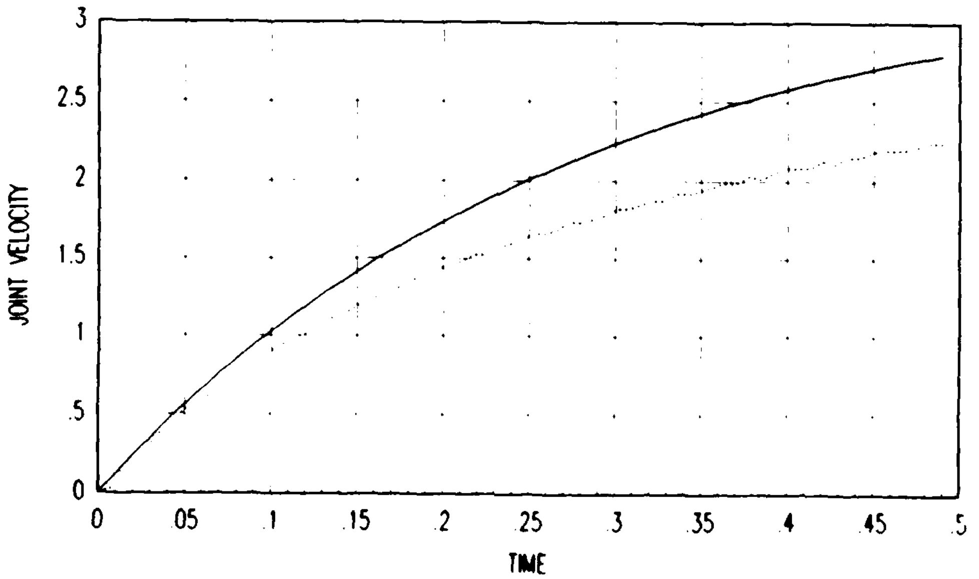


Figure E.14. Joint Six Step Response, Counts = 300

Ideal Model ......
Actual Response ______

Appendix F

SIMSTAR Hooks and Handles

1. Watch out for using too many analog components. If you use too many, rearrange the equation and try again. It may work.

2. If you have to reduce the model, begin with the coriolis terms. They have the least affect.

3. The PREPAR function in SIMRUN can not retrieve analog variables. It sometimes produces erroneous data. This is a function of the lack of a DCP on the AFIT SIMSTAR.

4. If you have compilation errors in FORTRAN 77 that don't make sense, look for a more obvious error earlier in the subroutine. Make sure you don't go past column 72. If you can't find an error, delete the line to which the error points and retype it.

5. Pick Don Smith's brain, EAI employee, if he is still around. Also, can call Rich Giddons at EAI for information.

6. Use ACSL to compile your program first. It will give you insight into scaling analog variables.

Bibliography

1. An, Chae H., Christopher G. Atkeson, John D. Griffiths, and John M. Hollerbach. "Experimental Evaluation of Feedforward and Computed Torque Control," Proceedings of the 1987 IEEE International Conference on Robotics and Automation. 169-174. Washington D.C., Computer Society Press of the IEEE. 1987

2. Goor, Robert M. "A New Approach to Minimum Time Robot Control", Robotics and Manufacturing Automation. pages 1-11. New York. presented to The Winter Annual Meeting of the American Society of Mechanical Engineers, 1985

3. Goor, Robert M. "A New Approach to Robot Control, Proceedings of the American Control Conference. pages 385-389. Boston, Ma. 1985

4. Hollerbach, John M. "A Recursive Lagrangian Formulation of Manipulator Dynamics and a Comparative Study of Dynamics Formulation Complexity," IEEE Transactions on Systems, Man, and Cybernetics, SMC-10, 730-736 IEEE Press (November 1980)

5. Khosla, Pradeep K. "Choosing Sampling Rates For Robot Control," Proceedings of the 1987 IEEE International Conference on Robotics and Automation. 169-174. Washington D.C., Computer Society Press of the IEEE. 1987

6. Koren, Yoram. Robotics for Engineers. New York. McGraw-Hill Book Company, 1985.

7. Leahy, Michael B., Jr. Robotic Manipulator Control Performance Evaluation, PhD Dissertation, Robotics and Automation Laboratory; Electrical, Computer and Systems Engineering Department; Rensselaer Polytechnic Institute, Troy, New York, August 1986, RAL #84

8. Leahy, Michael B., Jr. and G.N. Saridis. "Compensation of Unmodeled PUMA Manipulator Dynamics," Proceedings of the 1987 IEEE International Conference on Robotics and Automation. 151-156, Washington D.C., Computer Society Press of the IEEE. 1987

9. Lee, K.M. and D.K. Shaw, "Kinematic Analysis of a Three Degree of Freedom in Parallel Manipulator," Proceedings of the 1987 IEEE International Conference on Robotics and Automation. 345-350. Washington D.C., Computer Society Press of the IEEE. 1987 1987

10. Sweet, Larry M. and Malcolm C. Good. "Redefinition of the Robot Motion-Control Problem," 0272-1708/85/0800-0018, 1985, IEEE Press

11. Tarn, T.J., Bejczy, A.K., Han, Shuotiao, and Yun, Xiaoping. "Inertia Parameters of PUMA 560 Robot Arm," Robotics Laboratory Report #SSM-RL-85-01, Department of Systems Science and Mathematics, Washington University, St. Louis, Missouri, September 1985

VITA

Captain Peter Madison Van Wirt was born on 3 December 1960 in Charleston, West Virginia. He graduated from Archbishop Shaw High School, Marrero, Louisiana in 1979. He received the degree of Bachelor of Science in Electrical Engineering from Clemson University in 1983. Upon graduation, he received a commission in the USAF through the ROTC program. He was a project engineer for the North Warning System Program Office, Electronic Systems Division, Air Force Systems Command from 1984 to 1986. He entered the Air Force Institute of Technology in May 1986.

Captain Van Wirt and his wife Debra are the parents of two children: Stephanie, 2 yrs; and Michael, 8 wks.

UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE	*Form Approved OMB No. 0704-0188*
1a. REPORT SECURITY CLASSIFICATION UNCLASSIFIED	1b. RESTRICTIVE MARKINGS NONE
2a. SECURITY CLASSIFICATION AUTHORITY	3. DISTRIBUTION/AVAILABILITY OF REPORT Approved for Public Release; Distribution Unlimited
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE	
4. PERFORMING ORGANIZATION REPORT NUMBER(S) AFIT/GE/ENG/87D-68	5. MONITORING ORGANIZATION REPORT NUMBER(S)
6a. NAME OF PERFORMING ORGANIZATION AIR FORCE INSTITUTE OF TECHNOLOGY EN — 6b. OFFICE SYMBOL *(If applicable)* EN	7a. NAME OF MONITORING ORGANIZATION
6c. ADDRESS *(City, State, and ZIP Code)* AFIT/EN WRIGHT-PATTERSON AFB, OHIO 45433	7b. ADDRESS *(City, State, and ZIP Code)*
8a. NAME OF FUNDING/SPONSORING ORGANIZATION AFWAL/MLTZ — 8b. OFFICE SYMBOL *(If applicable)* AFWAL/MLTZ	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER
8c. ADDRESS *(City, State, and ZIP Code)* Wright-Patterson AFB, Ohio 45433	10. SOURCE OF FUNDING NUMBERS PROGRAM ELEMENT NO. / PROJECT NO. / TASK NO. / WORK UNIT ACCESSION NO.

11. TITLE *(Include Security Classification)*
Development of a Hybrid Simulator for Robotic Manipulators UNCLASS

12. PERSONAL AUTHOR(S)
Peter M. Van Wirt, B.S.E.E., Captain, USAF

13a. TYPE OF REPORT	13b. TIME COVERED	14. DATE OF REPORT *(Year, Month, Day)*	15. PAGE COUNT
THESIS	FROM ______ TO ______	1987 December	112

16. SUPPLEMENTARY NOTATION

17. COSATI CODES			18. SUBJECT TERMS *(Continue on reverse if necessary and identify by block number)*
FIELD	GROUP	SUB-GROUP	Robotic Control
			Robotic Simulation

19. ABSTRACT *(Continue on reverse if necessary and identify by block number)*

Sponsoring Organization: Computer Integrated Manufacturing Branch
Air Force Materials Laboratory

Thesis Advisor: Michael Leahy, Jr., Captain, USAF

Abstract: See Back

Approved for public release: IAW AFR 190-1.
[illegible] 14 May 88
LYNN E. WOLAVER
Dean for Research and Professional Development
Air Force Institute of Technology (AFC)
Wright Patterson AFB OH 45433

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT ☒ UNCLASSIFIED/UNLIMITED ☐ SAME AS RPT. ☐ DTIC USERS	21. ABSTRACT SECURITY CLASSIFICATION UNCLASS
22a. NAME OF RESPONSIBLE INDIVIDUAL Michael Leahy, Jr., Captain, USAF	22b. TELEPHONE *(Include Area Code)* 513/255/6027 — 22c. OFFICE SYMBOL AFIT/ENG

DD Form 1473, JUN 86 *Previous editions are obsolete.* SECURITY CLASSIFICATION OF THIS PAGE
UNCLASSIFIED

UNCLASSIFIED

19. Abstract. A real-time robot manipulator simulation capability has been developed. By programming the robot dynamics in the analog section of a SIMSTAR Hybrid Computer, the computational burden of digital integration techniques is avoided, and due to the analog nature of the model, the simulation can be run in real-time without sacrificing accuracy. The ability to test analog and hybrid control schemes is also achieved through the development of an analog manipulator model on the SIMSTAR and because the SIMSTAR is both a digital and analog computer. A hybrid controller contains an analog feedback portion to provide needed loop stiffness and a digital feedforward portion to compensate for the changing dynamics of a robotic manipulator. The model is developed through a combination of previous research and experimental evaluation. Once programmed, the SIMSTAR model is validated using known trajectory/error data obtained from the AFIT PUMA 560.

END
DATE
FILMED
DTIC
JULY 88